МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

Адашевская И. Ю.

ИНФОРМАЦИОННЫЕ СИСТЕМЫ КОНСТРУИРОВАНИЯ И МОДЕЛИРОВАНИЯ ОБЪЕКТОВ

Учебное пособие
для студентов и аспирантов специальности
«Информационные технологи проектирования»,
студентов технических специальностей, в том числе для иностранных студентов

Утверждено
редакционно-издательским
советом университета,
протокол № 2 від 23. 06. 16.

Харьков
НТУ «ХПИ»
2016

ББК 32.973.26-018.2**я**73
А28
УДК 004.92(075.8)

Рецензенты:

***Л. Н. Куценко**, д-р техн. наук, проф., Национальный Университет гражданской защиты Украины,*
***Ю.М. Тормосов**, д-р техн. наук, проф., Харьковский государственный университет питания и торговли.*

А28 **Адашевская И. Ю.**
Інформационные системы конструирования и моделирования объектов : учеб. пособие / И. Ю. Адашевская. – Харьков : НТУ «ХПИ», 2016. – 178 с. – На рус. яз.

ISBN 978-617-578-204-0

Представлено класифікацію інформаційних систем конструювання та моделювання об'єктів , надано аргументацію оптимального вибору пакета комп'ютерних програм для розв'язання різних завдань курсового і дипломного проектування , робіт з дизайну , їх порівняльні характеристики та приклади використання .

Призначено для студентів і аспірантів спеціальності « Інформаційні технології проектування», « Прикладна механіка » та інших технічних спеціальностей вищих навчальних закладів та для іноземних студентів.

В работе представлена клиссификация информационных систем конструирования и моделирования объектов, дана аргументация оптимального выбора пакета компьютерных программ для решения различных задач курсового и дипломного проектирования , работ по дизайну, их сравнительная характеристика и примеры использования.

Учебное пособие предназначено для студентов и аспирантов специальности «Информационные технологи проектирования», «Прикладная механика» и других технических специальностей высших учебных заведений.

Ил. 197 . Табл. 4 . Библиограф.: 55 назв.

ISBN 978-617-578-204-0 **ББК** 32.973.26-018.2**я**73

СОДЕРЖАНИЕ

Введение

Современный этап развития человечества отличается тем, что на смену века энергетики приходит век информатики. Происходит интенсивное внедрение новых информационных технологий во все сферы человеческой деятельности.

Методами компьютерного проектирования и моделирования пользуются специалисты практически всех отраслей и областей науки и техники от истории до космонавтики, поскольку с их помощью можно прогнозировать и даже имитировать явления, события или проектируемые предметы в заранее заданных параметрах.

Необходимость приоритетного развития образования в настоящее время обусловлена научно-техническим прогрессом и глобальной технологизацией передовых стран мира. Уровень современного производства, науки и техники, а также социальные преобразования определяют заинтересованность общества в подготовке конкурентоспособного, высококвалифицированного, интеллектуального и инициативного специалиста с развитым творческим мышлением.

Сложные по внутренним связям и объёмные по количеству элементов системы экономически трудно поддаются прямым способам проектирования и зачастую для построения и изучения переходят к имитационным методам. Появление новейших информационных технологий увеличивает не только возможности систем проектирования и моделирования, но и позволяет применять большее многообразие моделей и способов их реализации. Совершенствование вычислительной и телекоммуникационной техники привело к дальнейшему развитию методов проектирования и моделирования при помощи компьютерных систем, без которых невозможно изучение процессов и явлений, а также построение больших и сложных систем.

Конструирование и моделирование в научных исследованиях стало применяться еще с давних времён и постепенно захватывало все новые области научных знаний: техническое конструирование, строительство и архитектуру, астрономию, физику, химию, биологию и, наконец, общественные науки. Однако методология моделирования долгое время развивалась независимо в отдельных науках. Отсутствовала единая система понятий, единая терминология. Лишь постепенно стала осознаваться роль моделирования как универсального метода научного познания. Термин «модель» широко используется в различных сферах человеческой деятельности и имеет множество смысловых значений.

Актуальность данной проблемы - ее недостаточная разработанность. Применение компьютеров в научных исследованиях является необходимым условием изучения сложных систем. Традиционная методология взаимосвязи теории и эксперимента должна быть дополнена принципами компьютерного проектирования и моделирования. Эта новая эффективная процедура дает возможность целостного изучения поведения наиболее сложных систем, как естественных, так и создаваемых для проверки теоретических гипотез. Динамическая система представляет собой математическую модель некоторого объекта, процесса или явления.

Моделирование (в широком смысле) является основным методом исследований во всех областях знаний и научно обоснованным методом оценок характеристик сложных систем, используемым для принятия решений в различных сферах инженерной деятельности. Существующие и проектируемые системы можно эффективно исследовать с помощью математических моделей (аналитических и имитационных), реализуемых на современных ЭВМ, которые в этом случае выступают в качестве инструмента экспериментатора с моделью системы.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МЕТОДОВ КОНСТРУИРОВАНИЯ И МОДЕЛИРОВАНИЯ

1.1. Моделирование как метод научного познания

1.1.1 Теоретические основы моделирования

В настоящее время нельзя назвать область человеческой деятельности, в которой в той или иной степени не использовались бы методы моделирования. Особенно это относится к сфере управления различными системами, где основными являются процессы принятия решений на основе получаемой информации.

Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам; такие логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие природу явлений, называются моделями. Другими словами, модель (лат. niodulus – мера) – это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

Моделирование — замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели. Таким образом, моделирование может быть определено как представление объекта моделью для получения информации об этом объекте путем проведения экспериментов с его моделью. Теория замещения одних объектов (оригиналов) другими объектами (моделями) и исследования свойств объектов на их моделях называется теорией моделирования.

Определяя роль теории моделирования, т.е. ее значение, необходимо, прежде всего, отвлечься от имеющегося в науке и технике многообразия моделей и выделить то общее, что присуще моделям различных по своей природе объектов реального мира. Это общее заключается в наличии некоторой структуры (статической или динамической, материальной или мысленной), которая подобна структуре данного объекта. В процессе изучения модель выступает в роли относительного самостоятельного квазиобъекта, позволяющего получить при исследовании некоторые знания о самом объекте.

Если результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов, протекающих в исследуемых объектах, то говорят, что модель адекватна объекту. При этом адекватность модели зависит от цели моделирования и принятых критериев.

Обобщенно моделирование можно определить как метод опосредованного познания, при котором изучаемый объект-оригинал находится в некотором соответствии с другим объектом-моделью, причем модель способна в том или ином отношении замещать оригинал на некоторых стадиях познавательного процесса.

Стадии познания, на которых происходит такая замена, а также формы соответствия модели и оригинала могут быть различными:

1. Моделирование как познавательный процесс, содержащий переработку информации, поступающей из внешней среды, о происходящих в ней явлениях, в результате чего в сознании появляются образы, соответствующие объектам.

2. Моделирование, заключающееся в построении некоторой системы-модели (второй системы), связанной определенными соотношениями подобия с системой-оригиналом (первой системой), причем в этом случае отображение одной системы в другую является средством выявления зависимостей между двумя системами, отраженными в соотношениях подобия, а не результатом непосредственного изучения поступающей информации.

Процесс моделирования предполагает наличие объекта исследования; исследователя, перед которым поставлена конкретная задача; модели, создаваемой для получения информации об объекте и необходимой для решения поставленной задачи. Причем по отношению к модели исследователь является, по сути дела, экспериментатором, только в данном случае эксперимент проводится не с реальным объектом, а с его моделью. Такой эксперимент для инженера есть инструмент непосредственного решения организационно-технических задач.

1.2. **Использование моделирования при исследовании и проектировании сложных систем**

Одна из проблем современной науки и техники разработка и внедрение в практику проектирования новейших методов исследования характеристик сложных информационно-управляющих и информационно-вычислительных систем различных уровней (например: автоматизированных систем научных исследований и комплексных испытаний, систем автоматизации проектирования, комплексов и сетей, информационных систем). При проектировании сложных систем и их подсистем возникают многочисленные задачи, требующие оценки количественных и качественных закономерностей процессов функционирования таких систем, проведения их структурного алгоритмического и параметрического синтеза.

В дисциплине рассматриваются системы информатики и вычислительной техники, автоматизированные системы обработки информации и управления, информационные системы относятся к классу больших систем, этапы проектирования, внедрения, эксплуатации и эволюции которых в настоящее время невозможны без использования различных видов моделирования. На всех перечисленных этапах для сложных видов различных уровней необходимо учитывать такие особенности:

- сложность структуры и стохастичность связей между элементами, неоднозначность алгоритмов поведения при различных условиях;
- большое количество параметров и переменных, неполноту и недетерминированность исходной информации;
- разнообразие и вероятностный характер воздействий внешней среды.

Ограниченность возможностей экспериментального исследования больших систем делает актуальной разработку методики их моделирования, которая позволила бы в соответствующей форме представить процессы функционирования систем, описание протекания этих процессов с помощью математических моделей, получение результатов экспериментов с моделями по оценке характеристики исследуемых объектов. Причем на разных этапах создания и использования перечисленных систем для всего многообразия входящих в них подсистем применение метода моделирования преследует конкретные цели, а эффективность метода зависит от того, насколько грамотно разработчик использует возможности моделирования. Независимо от разбиения конкретной сложной системы на подсистемы при проектировании каждой из них необходимо выполнять внешнее проектирование (макропроектирование) и внутреннее проектирование (микропроектирование). Так как на этих стадиях разработчик преследует различные цели, то и используемые при этом методы и средства моделирования могут существенно отличаться.

На стадии макропроектирования должна быть разработана обобщенная модель процесса функционирования сложной системы, позволяющая разработчику получить ответы на вопросы об эффективности различных стратегий управления объектом при его взаимодействии с внешней средой. Стадию внешнего проектирования можно разбить на анализ и синтез.

При анализе изучают объект управления, строят модель воздействий внешней среды, определяют критерии оценки эффективности, имеющиеся ресурсы, необходимые ограничения. Конечная цель стадии анализа — построение модели объекта управления для оценки его характеристик.

При синтезе на этапе внешнего проектирования решаются задачи выбора стратегии управления на основе модели объекта моделирования, т.е. сложной системы.

На стадии микропроектирования разрабатывают модели с целью создания эффективных подсистем. Причем используемые методы и средства моделирования зависят от того, какие конкретно обеспечивающие подсистемы разрабатываются: информационные, математические, технические, программные и т. д.

Выбор метода моделирования и необходимая детализация моделей существенно зависят от этапа разработки сложной системы. На этапах обследования объекта управления, например промышленного предприятия, и разработки технического задания на проектирование автоматизированной системы управления модели в основном носят описательный характер и преследуют цель наиболее полно представить в компактной форме информацию об объекте, необходимую разработчику системы.

На этапах разработки технического и рабочего проектов систем модели отдельных подсистем детализируются, и моделирование служит для решения конкретных задач проектирования, т.е. выбора оптимального по определенному критерию при заданных ограничениях варианта из множества допустимых. Поэтому в основном на этих этапах проектирования сложных систем используются модели для целей синтеза.

Целевое назначение моделирования на этапе внедрения и эксплуатация сложных систем — это проигрывание возможных ситуаций для принятия обоснованных и перспективных решений по управлению объектом. Моделирование (имитацию) также широко применяют при обучении и тренировке персонала автоматизированных систем управления, вычислительных комплексов и сетей, информационных систем в различных сферах. В этом случае моделирование носит характер деловых игр. Модель, реализуемая обычно на ЭВМ, воспроизводит поведение управляемого объекта и внешней среды, а люди в определенные моменты времени принимают решения по управлению объектом.

АСОИУ являются системами, которые развиваются по мере эволюции объекта управления, появления новых средств управления и.т.д. Поэтому при прогнозировании развития сложных систем роль моделирования очень высока, так как это единственная возможность ответить на многочисленные вопросы о путях дальнейшего эффективного развития системы и выбора из них наиболее оптимального.

1.3. **Перспективы развития методов и средств моделирования систем в свете новых информационных технологий**

В последние годы основные достижения в различных областях науки и техники неразрывно связаны с процессом совершенствования ЭВМ. Сфера эксплуатации ЭВМ — бурно развивающаяся отрасль человеческой практики, стимулирующая развитие новых теоретических и прикладных направлений. Ресурсы современной информационно-вычислительной техники дают возможность ставить и решать математические задачи такой сложности, которые в недавнем прошлом казались нереализуемыми, например моделирование больших систем.

Исторически первым сложился аналитический подход к исследованию систем, когда ЭВМ использовалась в качестве вычислителя по аналитическим зависимостям. Анализ характеристик процессов функционирования больших систем с помощью только аналитических методов исследования наталкивается обычно на значительные трудности, приводящие к необходимости существенного упрощения моделей либо на этапе их построения, либо в процессе работы с моделью, что может привести к получению недостоверных результатов.

Поэтому в настоящее время наряду с построением аналитических моделей большое внимание уделяется задачам оценки характеристик больших систем на основе имитационных моделей, реализованных на современных ЭВМ с высоким быстродействием и большим объемом оперативной памяти. Причем перспективность имитационного моделирования как метода исследования характеристик процесса функционирования больших систем возрастает с повышением быстродействия и оперативной памяти ЭВМ, с развитием математического обеспечения, совершенствованием банков данных и периферийных устройств для организации диалоговых систем моделирования. Это, в свою очередь, способствует появлению новых «чисто машинных» методов решения задач исследования больших систем на основе организации имитационных экспериментов с их моделями. Причем ориентация на автоматизированные рабочие места на базе персональных ЭВМ для реализации экспериментов с имитационными моделями больших систем позволяет проводить не только анализ их характеристик, но и решать задачи структурного, алгоритмического и параметрического синтеза таких систем при заданных критериях оценки эффективности и ограничениях.

Расширение возможностей моделирования различных классов больших систем неразрывно связано с совершенствованием средств вычислительной техники и техники связи. Перспективным направлением является создание для целей моделирования иерархических многомашинных вычислительных систем и сетей.

Конец XX столетия ознаменовался интенсивным развитием и внедрением информатики во все сферы жизни общества. Это проявилось в интенсивном совершенствовании средств вычислительной техники и техники связи, в появлении новых и в дальнейшем развитии существующих информационных технологий, а также в реализации прикладных информационных систем. Достижения информатики заняли достойное место в организационном управлении, в промышленности, в проведении научных исследований и в автоматизированном проектировании. Информатизация охватила и социальную сферу: образование, науку, культуру, здравоохранение.

Применительно к естественным и техническим наукам принято различать такие виды моделирования:

- концептуальное моделирование, при котором совокупность уже известных фактов или представлений относительно исследуемого объекта или системы истолковывается с помощью некоторых специальных знаков, символов, операций над ними или с помощью естественного или искусственного языков;
- физическое моделирование, при котором модель и моделируемый объект представляют собой реальные объекты или процессы единой или различной физической природы, причем между процессами в объекте-оригинале и в модели выполняются некоторые соотношения подобия, вытекающие из схожести физических явлений;
- структурно-функциональное моделирование, при котором моделями являются схемы (блок-схемы), графики, чертежи, диаграммы, таблицы, рисунки, дополненные специальными правилами их объединения и преобразования;
- математическое (логико-математическое) моделирование, при котором моделирование, включая построение модели, осуществляется средствами математики и логики;
- имитационное (программное) моделирование, при котором логико-математическая модель исследуемого объекта представляет собой алгоритм функционирования объекта, реализованный в виде программного комплекса для компьютера.

Разумеется, перечисленные выше виды моделирования не являются взаимоисключающими и могут применяться при исследовании сложных объектов либо одновременно, либо в некоторой комбинации. Кроме того, в некотором смысле концептуальное и, скажем, структурно-функциональное моделирование неразличимы между собой, так как те же блок-схемы, конечно же, являются специальными знаками с установленными операциями над ними.

2. МЕТОДЫ МОДЕЛИРОВАНИЯ И КОНСТРУИРОВАНИЯ ОБЪЕКТОВ

Разработка программного обеспечения – это сложный процесс, при котором учитываются разнообразные критерии качества. Например, часто при конструировании программных объектов ориентируются на поддержку эволюционного расширения кода в ходе последующих доработок или повторное использование разработанных компонентов. Достижение таких критериев определяется как использованием соответствующей дисциплины программирования, так и применением языковых и инструментальных средств, ориентированных на поддержку избранной парадигмы.

Каждый из стилей программирования определяет свое видение базовых конструкций и создаваемых из них композиций. Например, при разработке с применением процедурного подхода программа разбивается на процедуры и данные. Применение объектно-ориентированного программирования (ООП) ведет к объединению процедур и данных в классы. Дополнительная инструментальная поддержка в ООП наследования и полиморфизма в сочетании с дисциплиной кодирования, опирающейся на позднее связывание, повышают гибкость процесса разработки и обеспечивают лучшую, по сравнению с процедурным подходом, поддержку эволюционного расширения кода и его повторного использования. Вместе с тем следует отметить, что методы разработки программного обеспечения постоянно расширяются, обеспечивая инструментальную поддержку новых критериев. Так, аспектно-ориентированное программирование поддерживает сквозное подключение новой функциональности в существующие классы без дополнительной модификации.

Подобное многообразие порождает интерес к исследованию того, какие методы и приемы используются для конструирования программных объектов в каждой из существующих парадигм программирования. Насколько эти методы отличаются друг от друга? Можно ли подходы, применяемые при построении программ в одном стиле, успешно использовать при другом стиле программирования? Можно ли выделить обобщенные приемы построения программ и позиционировать их независимо от языковых и инструментальных средств? Ответы на эти вопросы позволят сформировать модель, позволяющую описать методы конструирования понятий, независимо от их семантических особенностей.

Ниже проводится анализ методов конструирования программных объектов, а сам процесс конструирования воспринимается как некоторый обобщенный алгоритм, состоящий из применения операций конструирования над базовыми программными объектами. Основной идеей является то, что, несмотря на разнообразие парадигм и создаваемых с их помощью программных объектов, можно выделить ряд приемов, применяемых для написания кода независимо от используемого контекста. Это позволит формализовать взгляд на процесс конструирования и зафиксировать типовые приемы, связать их с любой из технологий разработки программного обеспечения.

В общем случае под программным объектом понимается любая конструкция независимо от ее семантического значения. Понятие программного объекта различается в разных парадигмах и языках программирования. В процедурном подходе к ним относятся процедуры и абстрактные типы данных. В объектно-ориентированной парадигме программным объектом в общем случае является класс. В функциональной парадигме основным программным объектом является функция.

2.1. Способы описания процесса конструирования

Для описания программных объектов используются разнообразные модели. Известно, что наибольшей информативностью обладают графические представления. Среди них можно выделить диаграммы Вирта, модель «сущность-связь», диаграммы классов. Все эти методы позволяют отразить семантическую модель программы с помощью набора графических объектов и показать связи между ними. Но существующие модели ориентированы на статическое представле-

ние понятий. Они не отражают протекание самого процесса конструирования данных понятий. Для изучения этого процесса необходимо обеспечить описание его динамики, что позволит провести изучение того, как программные объекты создаются, удаляются и модифицируются в процессе написания кода.

Программные объекты формируются на основе базовых понятий, из которых образуются более сложные конструкции. При конструировании выделяются различные варианты отношений, среди которых можно выделить иерархические отношения и отношения на основе ассоциаций. Иерархические отношения в языках программирования реализуются за счет вложения одних объектов в другие. В свою очередь, ассоциации – это ссылочные отношения, при разработке программы такие отношения задаются в виде ссылок, указателей и именованных связей.

Для поддержки эволюционного расширения кода изменения вносятся в дополнительный файл, который обычно объединяется воедино с уже существующей конструкцией за счет общности имен или указателей в комбинации с дополнительными операциями, обеспечивающими динамическое связывание в ходе выполнения программы. В этом случае исходный файл может не модифицироваться, а окончательное формирование единой программы осуществляется за счет использования специальных инструментальных средств (например, препроцессора и компоновщика) или дополнительного кода, специально для этого написанного программистом.

В различных языках программирования широко применяются разнообразные формы древовидных структур: структуры в *C*, классы в *C++*, списки в *Lisp* и т.д. Выстраивание иерархических отношений между объектами является первой попыткой при их систематизации, поэтому данный способ представления данных наиболее прост в освоении и понимании. В качестве подтверждения можно привести структуру типов и классов языка программирования (см. рис. 2.1).

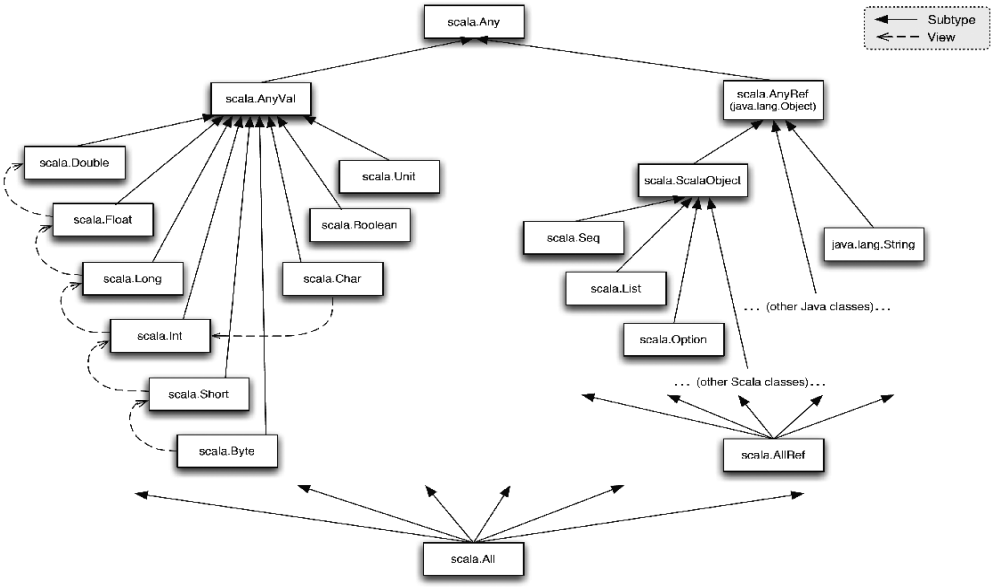


Рисунок 2.1 – Иерархия классов языка программирования *Scala*

2.2. **Анализ процесса конструирования программных объектов**

Модель конструирования программных объектов характеризуется статической составляющей, определяющей те структуры, которые получаются в результате конструирования, и динамической составляющей, описывающей набор операций, обеспечивающих построение структур.

Процесс конструирования программных объектов можно представить в виде набора «снимков», который показывает применение операций конструирования к статической структуре, полученной на предыдущем «снимке». Таким образом, процесс конструирования является некоторым процессом во времени, а создание программных объектов представляется действием, а не декларативным предписанием.

2.3. **Формальное описание модели конструирования объектов**

Описание модели конструирования объектов имеет следующую структуру:

- базовые объекты – набор исходных понятий. С точки зрения языка это базовые типы;
- операции конструирования – множество действий, выполняемых над базовыми объектами с целью получения новых структур;
- виды инструментальной поддержки. Определяют различные варианты использования операций конструирования;
- общее описание структуры, получаемое в результате применения операций конструирования.

Каждое определение нового объекта (или понятия) выглядит как некоторый набор уже определенных объектов (базовых или созданных программистом). Попробуем представить модель на простейшем уровне. Введем следующие тривиальные понятия: имя объекта, тип объекта, структура объекта. Как уже упоминалось, тип представляет собой ранее определенный объект, состоящий из других объектов и используемый при формировании структуры новых объектов. Тип сопоставляется с именем. Для более формализованного определения воспользуемся расширенными формами Бэкуса–Наура:

$$\$тип = \{/тип/\}, \tag{2.1}$$

$$\$тип = имя = содержание, \tag{2.2}$$

$$\$имя = \{/символ/\}, \tag{2.3}$$

$$\$содержание = имя|\textit{base}|структура, \tag{2.4}$$

$$\$структура = (\{/имя/\}). \tag{2.5}$$

Конструирование программных объектов – это создание абстракций, помогающих программировать в терминах целей решаемых задач, а не в терминах используемого языка программирования. Под программным объектом понимается сложная, относительно компонентов самостоятельная система, обладающая заданными свойствами и предназначенная для решения определеннойзадачи.Например:экранныеформы,меню,отчеты,удаленныепредставления(RemoteView в языке VisualFoxPro). Модель программного объекта – это высокоуровневое, декларативное описание, отображающее концептуальное представление разработчика об этом объекте. Наиболее удобной и адекватной формой такого описания является XML структура. Для использования модели в программе необходимо создать соответствующий генератор – механизм, синтезирующий

по спецификации модели выполняемый программный код. Разработанный подход можно представить в виде формулы:

[*Модель программного объекта*] + [*генератор*] = [*программный код*] (2.6)

Создание высокоуровневых моделей программных объектов согласуется с тенденцией повышения уровня языков программирования. Языки программирования навязывают свою собственную объектную модель и свой тип модульности, которые далеко не всегда адекватны логической модели задач, решаемых разработчиком. Работая с программой, разработчик строит свою собственную объектную модель как надстройку над моделью среды разработки. Эти и многие другие ограничения заставляют программиста думать как компьютер, вместо того чтобы научить компьютер понимать замыслы человека. Это серьезные, закоренелые ограничения и преодолеть их будет непросто. Ключевая идея данного подхода заключается в том, что предлагается не ждать, когда выйдет очередная новая версия языка программирования, а повышать уровень имеющегося в наличии. Причем повышать уровень не всего языка, а так называемыми «островками», создавая декларативные спецификации, отвечающие за реализацию часто встречающихся рутинных задач.

Мотивация использования высокоуровневых языковых расширений: нужна возможность работать в терминах концепций и понятий проблемы, которую необходимо решить, вместо того, чтобы переводить свои мысли в нотацию языка программирования общего назначения.

При разработке программ, особенно при разработке семейства программ, всегда можно выделить некие задачи, которые реализуются в разных программах из этого семейства или многократно реализуются в одной и той же программе. Предлагается выделять такие типовые, повторяющиеся задачи (например, построение экранных форм типа справочник, задание условий для отчетов, построение меню, выдача отчетов, работа с базой данных) и реализовывать их с помощью предлагаемого подхода. Необходимо отметить, что реализация данного подхода достаточно сложна и требует определенных знаний и навыков в программировании, поэтому реализовывать его следует только в том случае, когда его применение оправдывает силы, затраченные на его реализацию. Выделение таких задач требует определенного опыта и понимания предметной области.

Модель автоматизируемого программного объекта определяет набор тэгов и атрибутов создаваемой *XML* спецификации. Эта *XML* спецификация содержит как декларативную, так и процедурную часть, причем процедурная часть может быть вынесена в отдельный файл или же быть в одном файле с декларативной частью. Можно применять многоуровневый подход в построении *XML* спецификаций, т.е. получать одну *XML* модель на основе другой, тем самым повышая уровень декларативности задания объекта:

[*XML модель*] + [*генератор*] = [*XML модель*] + [*генератор*] = [*программ. код*] (2.7)

Алгоритм создания модели программного объекта:

1. Выбирается по некоторому критерию множество однотипных объектов;

2. Выделяется общая и специфичная части в интерфейсе и логике работы множества выделенных объектов;

3. Общая часть принимается за правило по умолчанию и интегрируется в генератор модели в качестве постоянной и неизменной части;

4. Специфичная часть параметризуется таким образом, чтобы она могла определять вид любого программного объекта из множества.

Свойства модели программного объекта:

- упрощённость. Использование модели объекта проще самого объекта, в том смысле, что программисту не нужно заботиться о «типичной» части объекта, которая принята за правило по умолчанию;
- достаточность. Несмотря на все умолчания, модель обладает достаточной гибкостью для описания необходимых объектов;
- массовость. Модель программного объекта позволяет описать не один конкретный объект, а целый набор однотипных программных объектов;
- повышенный уровень абстракции. Использование модели объекта предоставляет программисту более высокий уровень абстракции;
- параметризованность, позволяющая использовать различные сложные автовычисления и умолчания.

3. СИСТЕМЫ КОМПЬЮТЕРНОГО ПРОЕКТИРОВАНИЯ И МОДЕЛИРОВАНИЯ

Процесс проектирования и моделирования как средство познания действительности состоит из трех больших этапов:

- разработки математической модели и анализа разработанной модели;
- выбора метода решения и построения вычислительной установки по выбранной технологии решения;
- проведения вычислительного эксперимента (обработка и визуализация результатов эксперимента) и выявления закономерностей поведения реального объекта, явления или процесса.

В отрасли проектирования и моделирования реальных объектов условно выделились четыре направления: моделирование динамических систем, дискретно-событийное моделирование, агентное моделирование и системная динамика.

В соответствии с данными направлениями разрабатываются системы компьютерного проектирования и моделирования, которые условно можно подразделить на системы компьютерной математики, технического, графического и имитационного моделирования (рис. 3.1). Все эти системы развиваются, вносятся дополнения, и разработчики этих систем предлагают новые модернизированные версии. Изучить в полной мере все технологии достаточно сложно, однако знать об этих информационных системах и уметь использовать в своей профессиональной деятельности некоторые из них является необходимым условием компетентности специалиста в соответствующей области знаний.

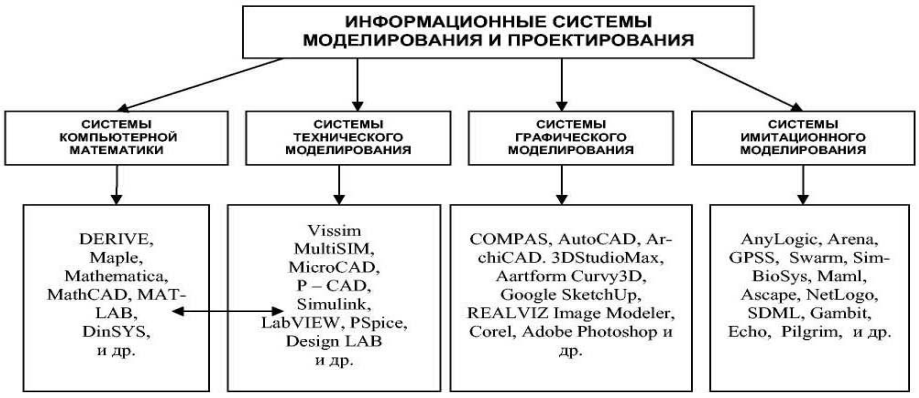


Рисунок 3.1 – Условная классификация информационных систем по типу решаемых задач

4. СИСТЕМЫ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

Эру создания компьютерной символьной математики принято отсчитывать с начала 60-х годов. Именно тогда в вычислительной технике возникла новая ветвь компьютерной математики, не совсем точно, но зато броско названная компьютерной алгеброй. Речь шла о возможности создания компьютерных систем, способных осуществлять типовые алгебраические преобразования: подстановки в выражениях, упрощение выражений, операции со степенными многочленами (полиномами), решение линейных и нелинейных уравнений и их систем, вычисление их корней и т. д. При этом предполагалась возможность получения аналитических (символьных) результатов везде, где это только возможно.

К сожалению, книги по этому направлению были способны лишь отпугнуть обычного читателя и пользователя компьютера от изучения возможностей компьютерной алгебры в силу перенасыщенности их узкоспециальным теоретическим материалом и весьма специфического языка описания. Материал таких книг, возможно, интересен математикам, занимающимся разработкой систем компьютерной алгебры, но отнюдь не основной массе их пользователей.

Большинство же пользователей заинтересовано в том, чтобы правильно выполнить конкретные аналитические преобразования, вычислить в символьном виде производную или первообразную заданной функции, разложить ее в ряд Тейлора или Фурье, провести аппроксимацию и т. д., а вовсе не в детальном и сложном математическом и логическом описании того, как это делается компьютером (или, точнее, его программистом). Здесь та же ситуация, что и с телевизором, радиоприемником или факсом: большинство из нас пользуются этими аппаратами, вовсе не интересуясь тем, как именно они выполняют свои довольно сложные функции.

Поняв эту истину, многие западные фирмы приступили к созданию компьютерных систем символьной математики, ориентированных на широкие круги пользователей, не являющихся профессионалами в компьютерной алгебре. Учитывая невероятно большую сложность автоматизации решения задач в аналитическом виде (число математических преобразований и соотношений весьма велико, и некоторые из них неоднозначны в истолковании), первые подобные системы удалось создать лишь для больших ЭВМ. Но затем появились и системы, доступные для мини-ЭВМ.

Системы компьютерной математики – новые средства, автоматизирующие выполнение как численных, так и аналитических вычислений. Они аккумулируют и предоставляют пользователю возможности, накопленные за многовековой опыт развития математики, имеют прекрасную цветную графику. Позволяют готовить электронные уроки и книги с живыми примерами и представляют большой интерес для системы образования.

В последнее время мы стали свидетелями появления нового, актуального и полезного научного направления – компьютерной математики. Ее можно определить как совокупность теоретических, алгоритмических, аппаратных и программных средств, предназначенных для эффективного решения на компьютерах всех видов математических задач с высокой степенью визуализации всех этапов вычислений. Последнее играет решающую роль во внедрении систем компьютерной математики (СКМ) в образование – какв высшее, так и начальное.

Системы компьютерной математики уже используются для решения учебных, научных и инженерных задач, наглядной визуализации данных и результатов вычислений и в качестве удобных и полных справочников по математическим вычислениям. Они стали мощным инструментом для подготовки электронных уроков, курсов лекций и электронных книг с живыми примерами, которые учащийся может менять.

Спрос на универсальные и специализированные программные пакеты для решения различных прикладных задач вызвал появление на рынке программных продуктов систем компьютерной математики, которые быстро стали популярными. На рынке современных математических систем

(рис. 4.1) в настоящее время присутствует целый ряд крупных фирм: Macsyma, Inc., Waterloo Maple Software, Inc., Wolfram Research, Inc., MathWorks, Inc., MathSoft, Inc., SciFace GmbH и др.

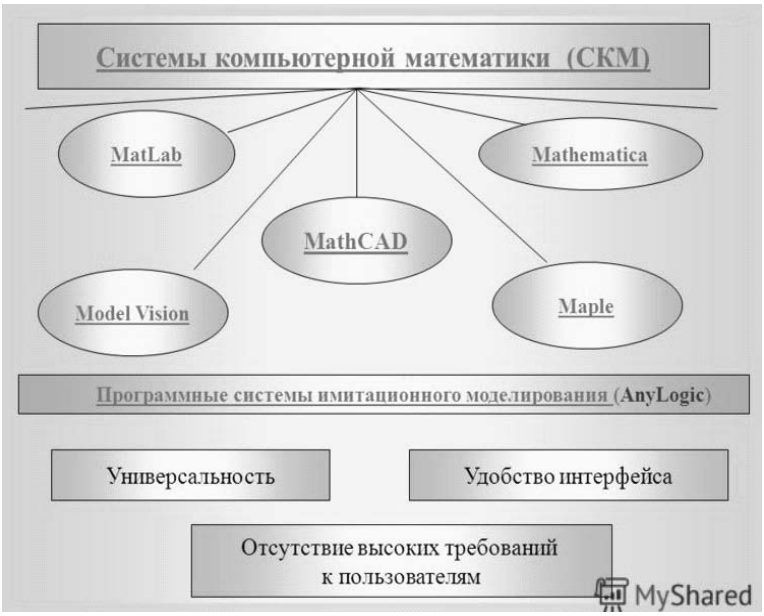


Рисунок 4.1 – Система компьютерной математики

К разработке каждой такой математической системы привлекаются сотни профессионалов из известных университетов и крупных научных центров, а также высококвалифицированные программисты и эксперты в области проектирования сложных программных систем. В результате мы имеем весьма совершенные, гибкие и одновременно универсальные продукты, включающие существенные математические понятия и обладающие богатым набором методов для решения общих математических и научно-технических задач.

Заметное развитие получили языки программирования для символьных вычислений *Reduce,* система *muMath* для малых ЭВМ, а в дальнейшем – интегрированные системы символьной математики для персональных компьютеров: *Derive, MathCAD, Mathematica, Maple V* и др.

4.1 **Конструирование объектов с использованием пакета** Mathematica

Mathematica – система компьютерной алгебры, используемая во многих научных, инженерных, математических и компьютерных областях. Изначально система была придумана Вольфрамом, в настоящее время разрабатывается компанией *Wolfram Research.*

Кроме того, *Mathematica* – это интерпретируемый язык функционального программирования. Можно сказать, что система *Mathematica* написана на языке *Mathematica*, хотя некоторые функции, особенно относящиеся к линейной алгебре, в целях оптимизации были написаны на языке С.

Mathematica поддерживает и процедурное программирование с применением стандартных операторов управления выполнением программы (циклы и условные переходы), и объектно-ориентированный подход. *Mathematica* допускает отложенные вычисления. Также в системе *Mathematica* можно задавать правила работы с теми или иными выражениями.

С помощью программы *Mathematica* можно осуществлять широкий спектр символьных преобразований, включающий наряду с другими операциями математического анализа: дифференцирование, интегрирование, разложение в ряды, решение дифференциальных уравнений и т.д. Одна из сильных сторон рассматриваемого программного продукта – развитая двумерная и трехмерная графика.

Программа состоит из двух частей (рис. 4.2): ядра математических операций (*Kernel*), которое производит вычисления, выполняя заданные команды, и интерфейсного процессора (*Front End*), который определяет, какой вид имеет пользовательский интерфейс программы в окне. Для расширения набора функций служит библиотека *Library* и набор пакетов расширения *Add-on Packages*.

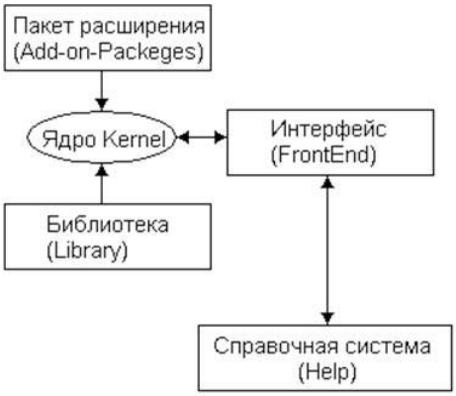


Рисунок 4.2 – Структура системы Mathematica

Возможности:

1. Аналитические преобразования:
   - решение систем полиномиальных и тригонометрических уравнений и неравенств, а также трансцендентных уравнений, сводящихся к ним;
   - решение рекуррентных уравнений;
   - упрощение выражения;
   - нахождение пределов;
   - интегрирование и дифференцирование функций;
   - нахождение конечных и бесконечных сумм и произведений;
   - решение дифференциальных уравнений и уравнений в частных производных;
   - преобразования Фурье и Лапласа, а также Z-преобразование;
   - преобразование функции в ряд Тейлора, операции с рядами Тейлора: сложение, умножение, композиция, получение обратной функции. и.т.д.;
   - вейвлет-анализ;
   - численные расчёты:
   - вычисление значений функций, в том числе специальных, с произвольной точностью;
   - решение систем уравнений;
   - нахождение пределов;
   - интегрирование и дифференцирование;
   - нахождение сумм и произведений;
   - решение дифференциальных уравнений и уравнений в частных производных;
   - полиномиальная интерполяция функции от произвольного числа аргументов по набору известных значений;

- преобразования Фурье и Лапласа, а также Z-преобразование;
- расчёт вероятностей.

2. Теория чисел:
   - определение простого числа по его порядковому номеру, определение количества простых чисел, не превосходящих данное;
   - дискретное преобразование Фурье;
   - разложение числа на простые множители, нахождение НОД и НОК.
3. Линейная алгебра:
   - операции с матрицами: сложение, умножение, нахождение обратной матрицы, умножение на вектор, вычисление экспоненты, получение определителя;
   - поиск собственных значений и собственных векторов.
4. Графика и звук:
   - построение графиков функций, в том числе параметрических кривых и поверхностей;
   - построение геометрических фигур: ломаных, кругов, прямоугольников, и т.д.;
   - воспроизведение звука, график которого задаётся аналитической функцией или набором точек;
   - импорт и экспорт графики во многих растровых и векторных форматах, а также звука;
   - построение и манипулирование графами;
   - разработка программного обеспечения;
   - автоматическое генерирование С-кода и его компоновка;
   - автоматическое преобразование компилируемых программ системы *Mathematica* в С-код для автономного или интегрированного использования;
   - использование *Symbolic C* для создания, обработки и оптимизации С-кода;
   - интеграция внешних динамических библиотек;
   - поддержка *CUDA* и *Open CL*;
   - язык программирования *Mathematica*.

Кроме того, *Mathematica* – это интерпретируемый язык функционального программирования. Можно сказать, что система *Mathematica* написана на языке *Mathematica*, хотя некоторые функции, особенно относящиеся к линейной алгебре, в целях оптимизации были написаны на языке *C*.

Mathematica поддерживает и процедурное программирование с применением стандартных операторов управления выполнением программы (циклы и условные переходы), и объектно-ориентированный подход. *Mathematica* допускает отложенные вычисления. Также в системе *Mathematica* можно задавать правила работы с теми или иными выражениями.

Примеры работы:

В начале запуска, сопровождаемого музыкальным звуком, *Mathematica* выводит чистое окно редактирования документа, в котором нет даже маркера ввода – характерной вертикальной черточки. Этот маркер появится, как только вы введете какой-то первый символ. После получения первого результата появляется и длинная горизонтальная черта, отделяющая выведенные ячейки от свободного поля окна редактирования под ними. Эта черта является признаком возможности ввода очередной ячейки. Ее можно перевести в уже созданную область документа, если вы захотите создать новую ячейку среди уже существующих ячеек ввода.

Обратите внимание на то, что система выделяет ячейки ввода определителем In [N], а ячейки вывода – определителем Out [N], где N – автоматически проставляемый номер строки.

Для решения уравнений (как одиночных, так и систем) в численном и символьном виде *Mathematica* имеет функцию *Solve*.

С функцией *Solve* можно использовать ряд опций. Их можно вывести командой *Options [Solve]*. Ниже описано их назначение:

– Inverse Functions – указывает, следует ли использовать обратные функции;

– Make Rules – указывает, должен ли результат быть представлен как объект AlgebraicRulesData;

– Method – устанавливает алгоритм, используемый для вычисления результата (возможны методы 1, 2 и 3);

– Mode – задает характер решения уравнения (возможны Generic, Modular и Rational);

– Sort – устанавливает, нужна ли сортировка результатов;

– Verif y Solutions – устанавливает, следует ли проводить проверку полученных решений и удаление посторонних решений;

– Working Precision – устанавливает число цифр промежуточных вычислений (по умолчанию Infinity).

Примеры (рис. 4.3, 4.4):

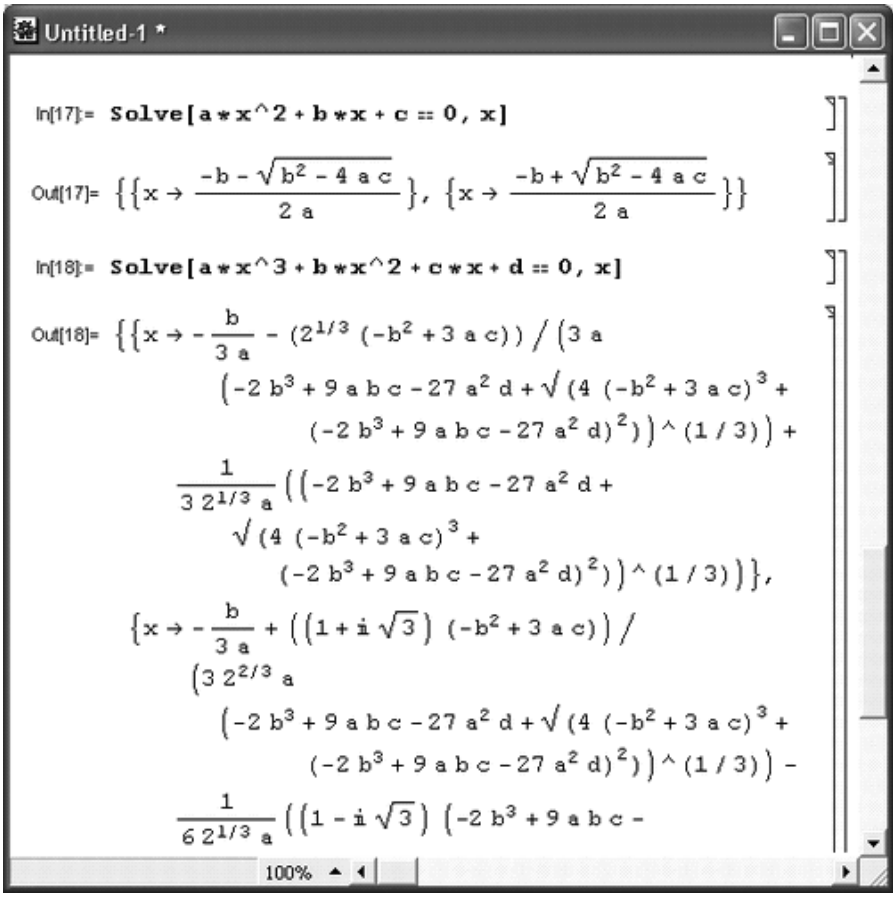


Рисунок 4.3 – Система *Mathematica* решает квадратное и кубическое уравнения

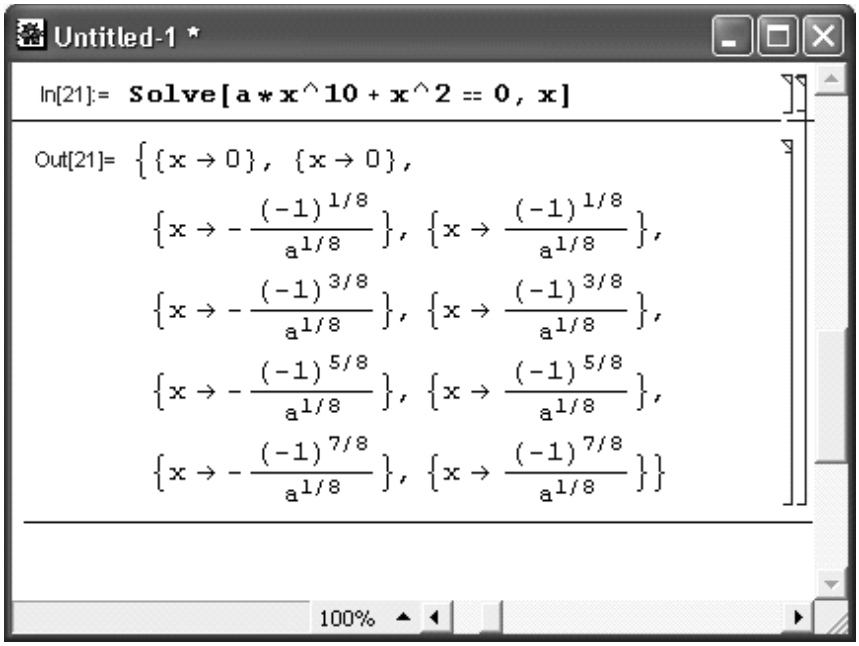


Рисунок 4.4 – Система *Mathematica* решает алгебраическое уравнение 10-й степени

Многие нелинейные уравнения и системы нелинейных уравнений в принципе не имеют аналитических решений. Однако их решение вполне возможно численными методами (рис. 4.5). Для численного решения систем нелинейных уравнений используется функция NSolve:

– NSolve [eqns, vars] – пытается численно решить одно уравнение или систему уравнений eqns относительно переменных vars;

– NSolve [eqns, vars, elims] – пытается численно решить уравнения eqns относительно vars, исключая переменные elims.

С этой функцией используется единственная опция Working Precision, задающая число верных цифр результата – по умолчанию 16.

Графическая иллюстрация и выбор метода решения уравнений.

При рассмотрении приведенных выше примеров может сложиться благодушное впечатление о том, что решение нелинейных уравнений может производиться автоматически и без размышлений. Но это далеко не так – представленные выше примеры просто подобраны так, что они имеют решение с помощью соответствующих функций.

На самом деле порой даже простые уравнения могут не иметь решения. В сложных случаях очень полезна графическая визуализация решения.

В качестве примера на рис. 4.6 ниже показана визуализация вычисления корней квадратного уравнения. В данном случае график функции (см.рис. 4.6) явно указывает на существование двух действительных корней при *x*, близких к 0,2 и 2,3. Функция Nsolve без труда находит оба корня.

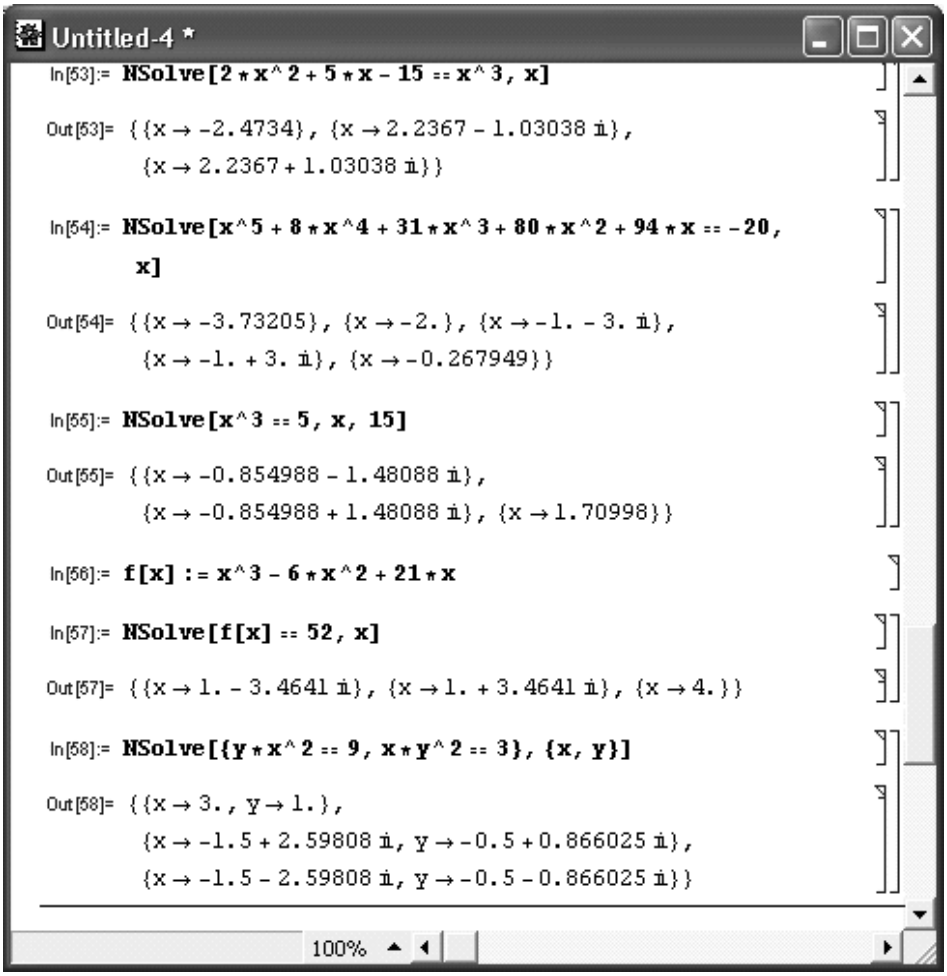


Рисунок 4.5 – Примеры численного решения уравнений

Получение сразу нескольких корней

Многие уравнения с тригонометрическими функциями могут иметь периодические или близкие к ним решения. К сожалению, функции *Mathematica*, вычисляющие корни уравнений, не способны в этом случае дать сразу несколько корней. Однако ситуация тут далеко не безнадежна – приведенный ниже пример наглядно показывает это.

Например, пусть требуется в интервале изменения *x* от 0 до 20 найти все решения уравнения:

$$x \sin(x) + x/2 — 1 = 0 \qquad (4.1)$$

График функции, представляющей левую часть уравнения, показан на рис. 4.7. Хорошо видно, что график пересекает ось *x* семь раз, то есть имеет в интересующем нас диапазоне семь корней.

На рис. 4.8 ниже показано построение графика функции *Sin [x] /x* ,заданной как функция пользователя, и ее производной с помощью функции Plot.

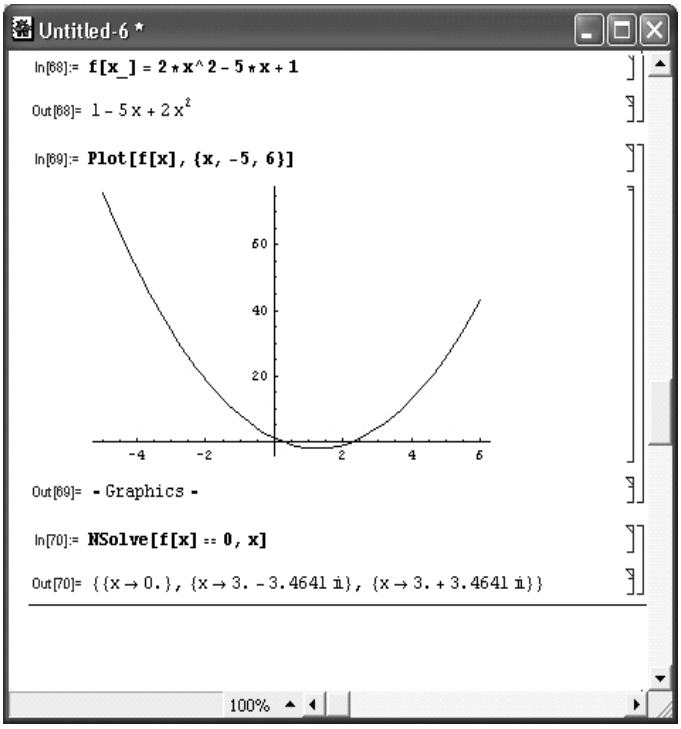


Рисунок 4.6 – Визуализация решения квадратного уравнения для случая двух действительных корней

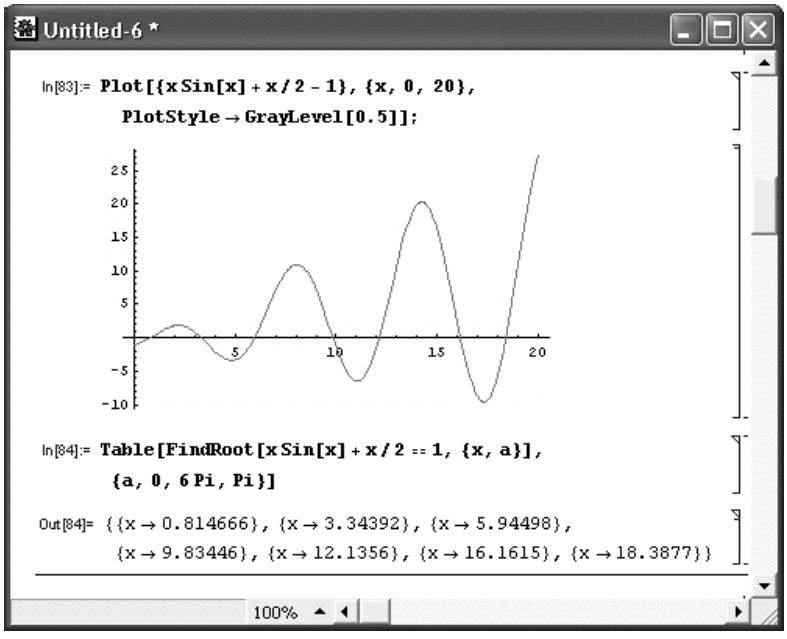


Рисунок 4.7 – График функции $x \sin(x) + x/2 - 1$ и пример вычисления всех ее корней в интервале изменения x от 0 до 20, где Plot – графическая функция.

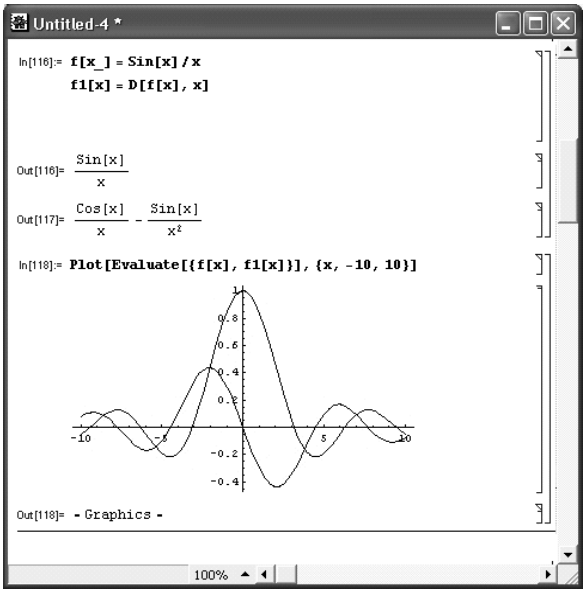


Рисунок 4.8 – График функции *Sin[x]/x* и ее производной

Примеры решений дифференциальных уравнений, построения различных поверхностей и фигур показаны на рис. 4.9 – 4.12.

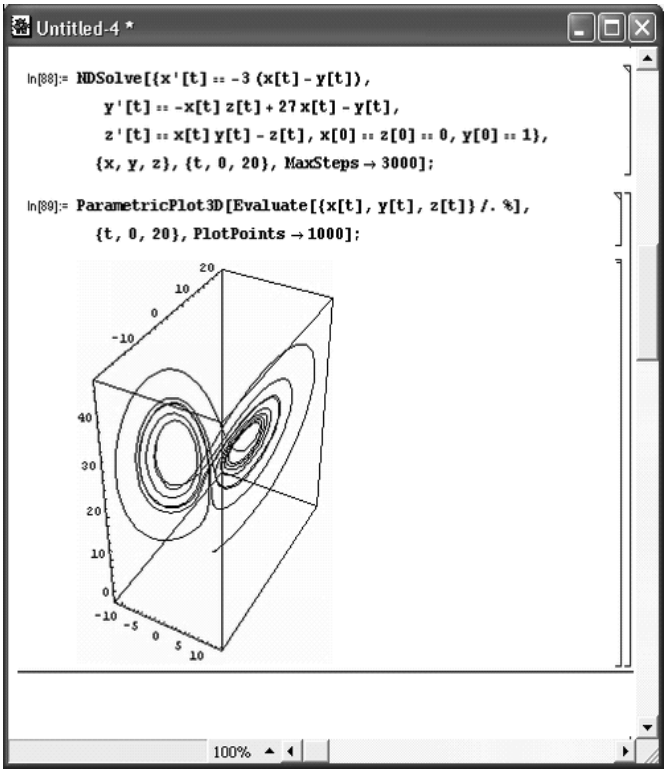


Рисунок 4.9 – Решение системы дифференциальных уравнений с выводом решения в форме кривых на фазовых плоскостях

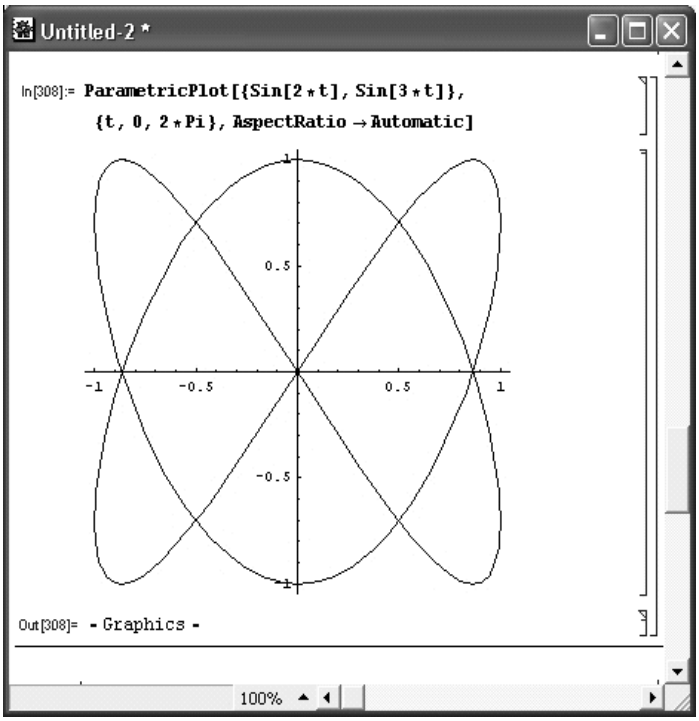


Рисунок 4.10 – Построение фигуры Лиссажу

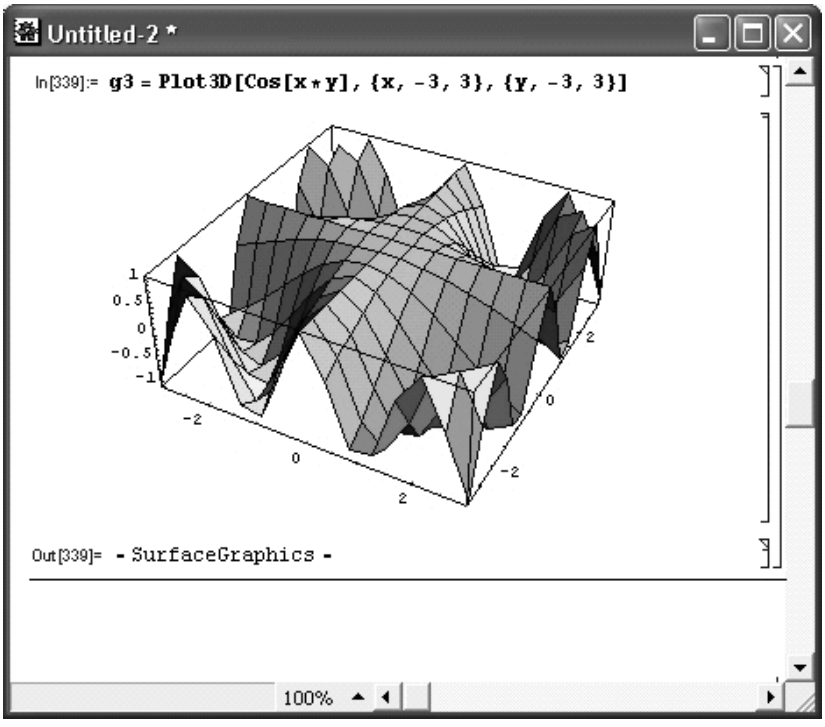


Рисунок 4.11 – Пример построения поверхности *cos(xy)* функцией Plot3D с опциями по умолчанию

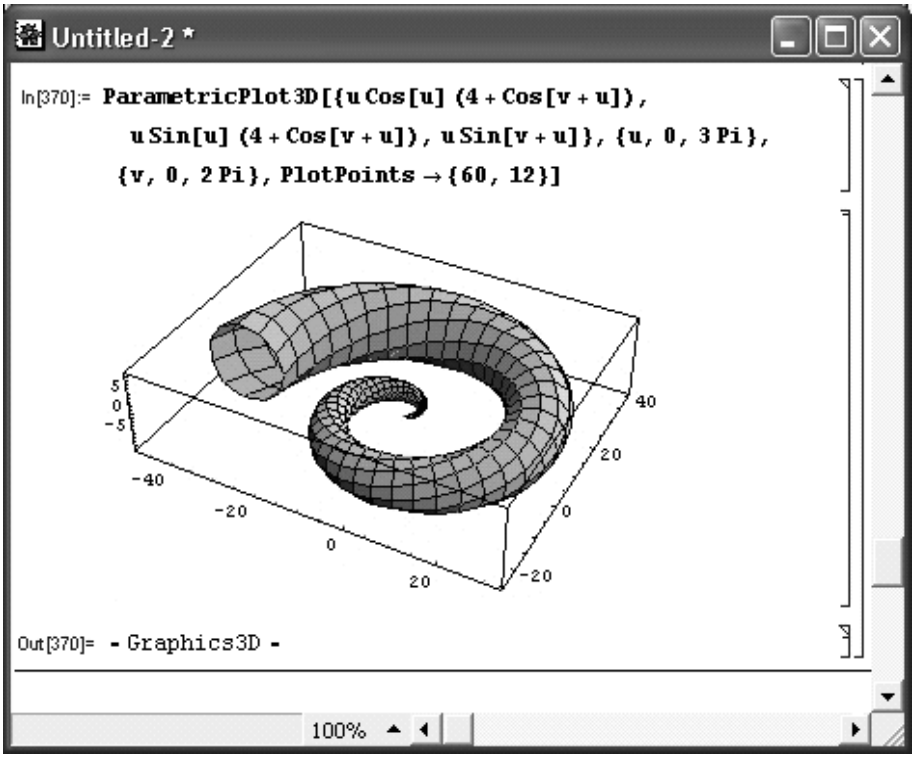


Рисунок 4.12 – Построение фигуры «рог»

Подпакет Polyhedra служит для создания регулярных пространственных фигур – полиэдров. Они задаются как графические примитивы и выводятся функцией Show:

– Show [Polyhedron [polyname]] – строит полиэдр с именем polyname в центре графика;

– Show [Polyhedron [polyname,{x,y,z},scale]] – строит полиэдр с именем polyname с центром в точке *{x, y, z}* и параметром масштаба scale.

Возможно задание следующих имен полиэдров: Tetrahedron, Cube, Octahedron, Dodecahedron, Icosahedron, Hexahedron, GreatDodecahedron, Small-StellatedDodecahedron, GreatStellatedDodecahedron и Greatlcosa-hedron. Пример построения полиэдра Icosahedron показан на рис. 4.13.

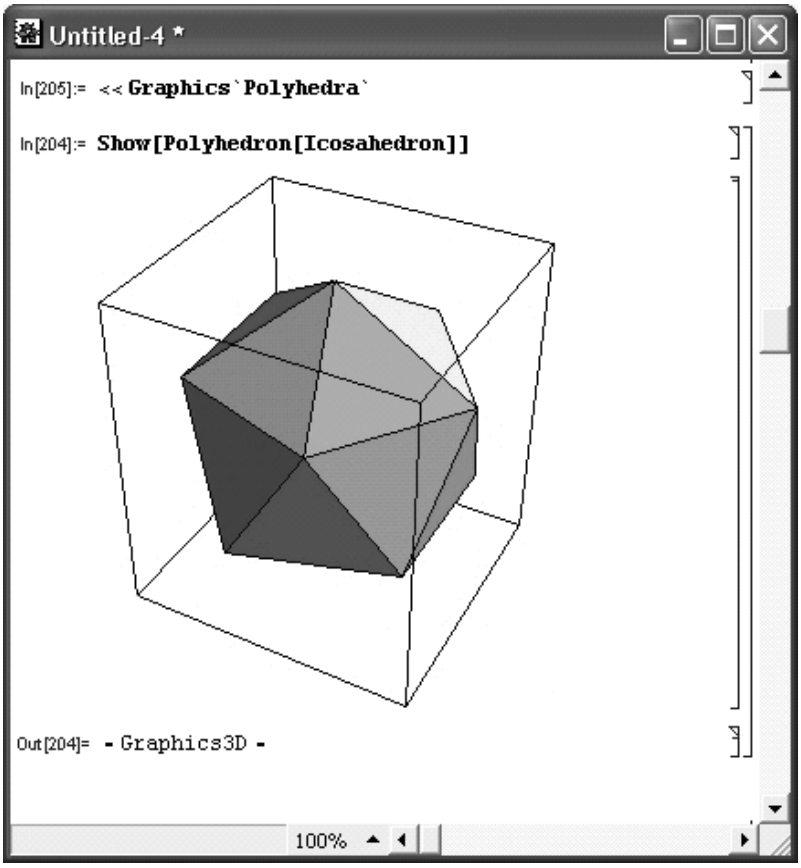


Рисунок 4.13 – Построение полиэдра Icosahedron

4.2. **Конструирование объектов с использованием пакета** Maple

Maple – система компьютерной математики, рассчитанная на широкий круг пользователей. До недавнего времени ее называли системой компьютерной алгебры, это указывало на особую роль символьных вычислений и преобразований, которые способна осуществлять эта система. Но такое название сужает сферу применения системы. На самом деле она уже способна выполнять быстро и эффективно не только символьные, но и численные расчеты, причем сочетает это с превосходными средствами графической визуализации и подготовки электронных документов.

Казалось бы, нелепо называть такую мощную систему, как *Maple* математической системой «для всех». Однако по мере ее распространения она становится полезной для многих пользователей ПК. Использование Maple простирается от решения учебных задач в вузах до моделирования сложных физических объектов, систем и устройств, и даже создания художественной графики (например, фракталов).

Вообще говоря, системы *Maple* ориентированы на решение сложных задач, хотя и решение в них простых задач вполне допустимо и уместно. Возможно, для решения таких задач вполне подойдет весьма простая, быстрая и надежная система *Derive* или система *Mathcad*, в которую (начиная с версии 3.0 под Windows) включен приобретенный по лицензии фирмы Waterloo Maple

упрощенный символьный процессор Maple. Однако по числу доступных пользователю математических функций эти скромные системы не идут ни в какое сравнение с патриархом символьной математики – системой *Maple*.

Система *Maple* может с успехом применяться для решения самых серьезных математических задач аэродинамики, теории поля, теплопроводности и диффузии, теоретической механики и др. Решение таких задач нередко является многолетним трудом элитных научных коллективов.

4.2.1. ***Построение двумерных графиков***

Maple реализует все мыслимые (и даже «немыслимые») варианты математических графиков. Строятся как графики простых функций в декартовой и полярной системах координат, так и графики, показывающие реалистические образы сложных, пересекающихся в пространстве фигур с их функциональной окраской. Возможны наглядные графические иллюстрации решений самых разнообразных уравнений, включая системы дифференциальных уравнений.

В само ядро Maple встроено ограниченное число функций построения графиков. Это, прежде всего, функция для построения двумерных графиков plot и функция для построения трехмерных графиков plotSd. Они позволяют строить графики наиболее распространенных типов. Для построения специальных графиков (например, векторных полей градиентов, решения дифференциальных уравнений, построения фазовых портретов и т. д.) в пакеты системы *Maple* включено большое число различных графических функций. Для их вызова необходимы соответствующие указания.

Графические функции заданы таким образом, что обеспечивают построение типовых графиков без какой-либо особой подготовки. Для этого нужно лишь указать функцию, график которой строится, и пределы изменения независимых переменных. Однако с помощью дополнительных необязательных параметров можно существенно изменить вид графиков – например, настроить стиль и цвет линий, вывести титульную надпись, изменить вид координатных осей и т. д.

В математике широко используются зависимости вида *y(x)* или *y(x)*. Их графики строятся на плоскости в виде ряда точек *y*1(*x*1), обычно соединяемых отрезками прямых. Таким образом, используется кусочно-линейная интерполяция двухмерных графиков. Если число точек графика достаточно велико (десятки или сотни), то приближенность построения не очень заметна.

Для построения двухмерных графиков служит функция *plot.* Она задается в виде:

$$plot(f, h, v), \tag{4.2}$$

$$plot\ (f, h, v, o), \tag{4.3}$$

где*: f* – визуализируемая функция (или функции),

h – переменная с указанием области ее изменения,

v – необязательная переменная с указанием области изменения,

o – параметр или набор параметров, задающих стиль построения графика (толщину и цвет кривых, тип кривых, метки на них и т. д.).

При построении графика одной функции (рис. 4.14, 4.15) она записывается в явном виде на месте шаблона *f.* Примеры построения графика одной функции даны ниже.

При построении графиков одной функции могут быть введены описание диапазонов и различные параметры, например: для задания цвета кривой, толщины линии, которой строится график функции, и др. К примеру, запись в списке параметров color = black задает вывод кривых черным цветом, а запись thikness = 2 задает построение графика линией, удвоенной по сравнению с обычной толщиной. Кстати говоря, запись color = red дает красный цвет, color = green – зеленый цвет, color = blue – синий цвет и т. д. При черно-белой печати цвета представляются оттенками серого цвета.

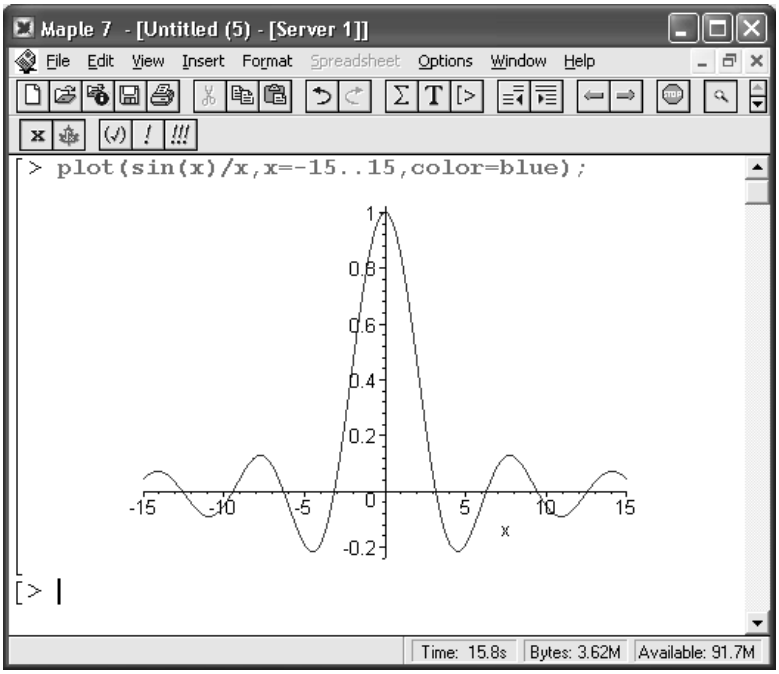


Рисунок 4.14

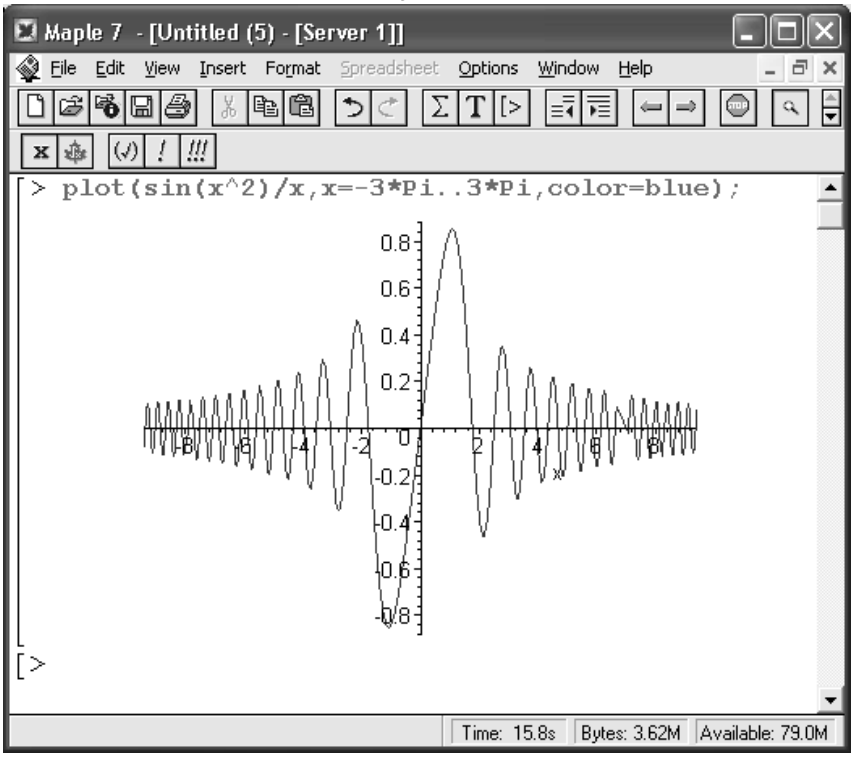


Рисунок 4.15

*4.2.2. **Двухмерные и трёхмерные графики***

Трехмерными называют графики, отображающие функции двух переменных *z(x,y)*. Каждая точка z_i таких графиков является высотой (аппликатой) точки, лежащей в плоскости XY и представленной координатами *(x,y).* Поскольку экран монитора компьютера в первом приближении является плоским, то на деле трехмерные графики представляют собой специальные проекции объемных объектов (рис. 4.16).

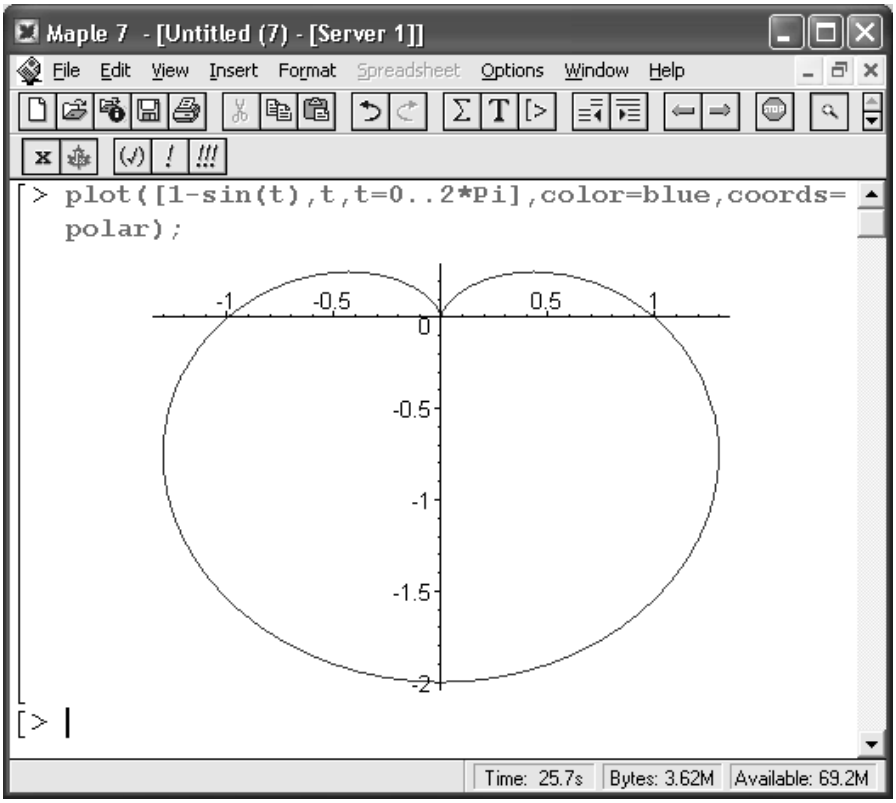


Рисунок 4.16 – Построение трехмерных графиков

Для построения графиков трехмерных поверхностей Maple имеет встроенную в ядро функцию plot3d. Она может использоваться в следующих форматах:

plot3d(exprl. x = a..b. y = c..d, p). (4.4)

plot3 d(f, a..b. c..d.p). (4.5)

plot3d([exprf.exprg.exprh]. s = a..b, t = c..d.p) (4.6)

plot3d([f.g.h]. a..b, c..d,p). (4.7)

В двух первых формах plot3d применяется для построения обычного графика одной поверхности, в других формах – для построения графика с параметрической формой задания поверхности. В приведенных формах записи *f, g* и *h* – функции; exprl – выражение, отражающее зависимость

от *x* и *y*; exprf, exprg и exprh – выражения, задающие поверхность параметрически; *s, t, a* и *b* – числовые константы действительного типа; end – числовые константы или выражения действительного типа; *x, y, s* и *t* – имена независимых переменных; p – управляющие параметры.

Несколько примеров построения графиков трехмерных поверхностей (рис. 4.17, 4.18):

1) Построение кардиоиды:

```
>plot([3*(1-cos(t)),t,t=0..2*Pi],coords=polar);
```

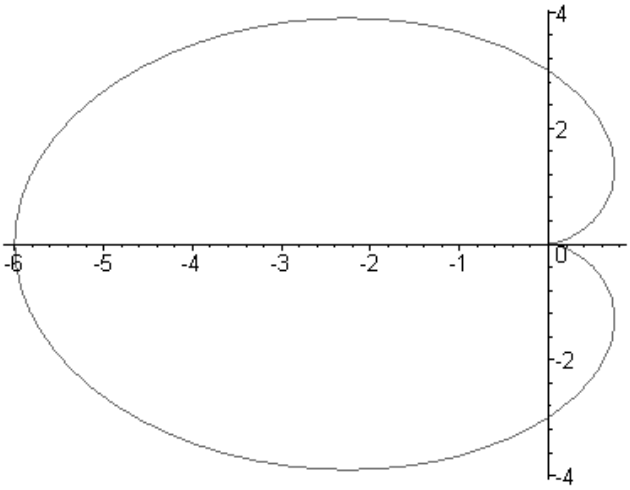


Рисунок 4.17 – Построение кардиоиды

2) Построение трехлепестковой розы:

```
>plot([2*sin(3*t),t,t=0..2*Pi],coords=polar) ;
```

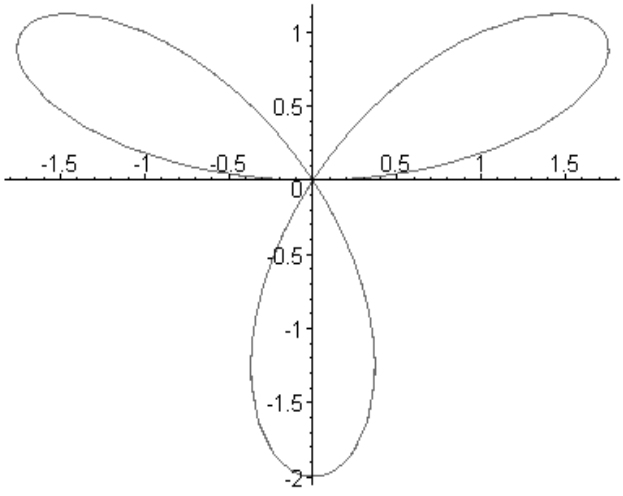


Рисунок 4.18 – Построение трехлепестковой розы

3) Построение поверхностей с разными стилями:

На рис. 4.19 показано пример простейшего построения графика трехмерной поверхности.

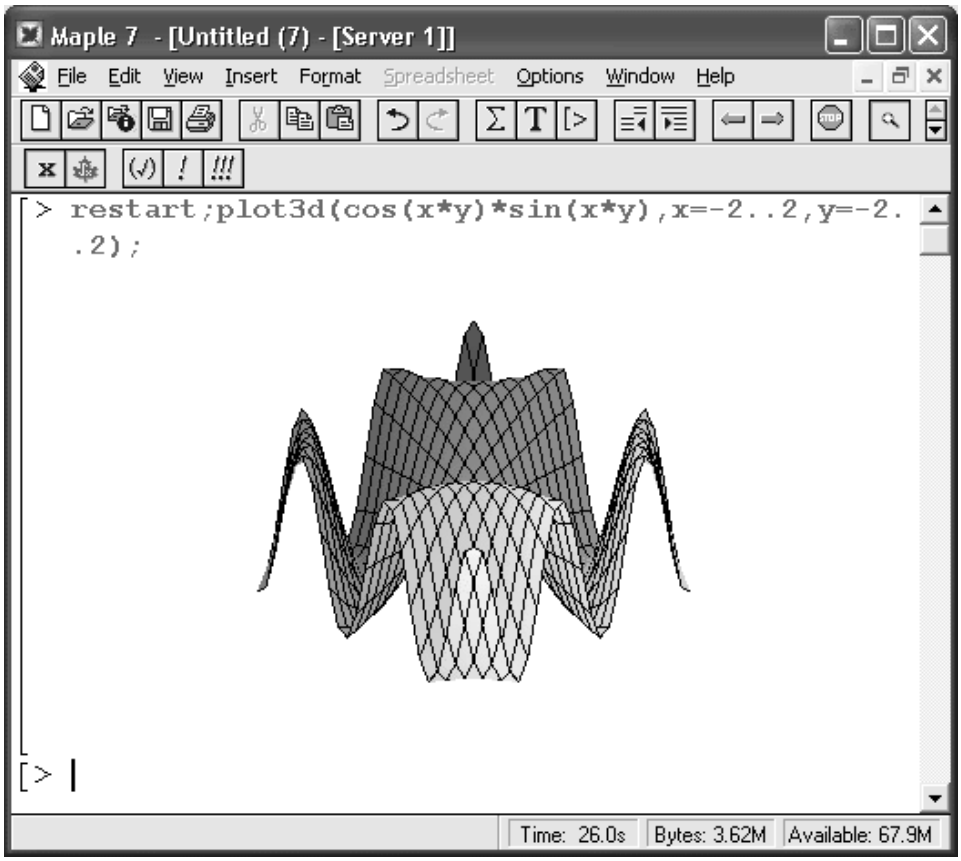


Рисунок 4.19 – Построения графика трехмерной поверхности

По умолчанию в *Maple* строится поверхность с функциональной окраской и стилем style = patch. Функциональная окраска делает рисунки более информативными. Параметр style = hidden строит каркасную поверхность с функциональной окраской тонких линий каркаса и удалением невидимых линий.

Помимо значения patch для построения трехмерных поверхностей можно задавать ряд других стилей: point – точками, contour – контурными линиями, line – линиями, hidden – линиями каркаса с удалением невидимых линий, wireframe – линиями каркаса со всеми видимыми линиями, patchnogrid – с раскраской, но без линий каркаса, patchcontour – раскраска с линиями равного уровня.

Цвет трехмерного графика может задаваться (как и для двухмерного) параметром color = c, где c –цвет.

Возможно еще два алгоритма задания цвета:

HUE – алгоритм с заданием цвета в виде color = f(x,y);

RGB – алгоритм с заданием цвета в виде color = [exprr,exprg,exprb], где выражения exprr, exprg и exprb задают относительную значимость (от 0 до 1) основных цветов (красного – exprr, зеленого – exprg и синего – exprb).

Удачный выбор углов обзора фигуры и применение функциональной окраски позволяют придать построениям трехмерных фигур весьма эффектный и достоверный вид.

Вид графика трехмерной поверхности существенно зависит от выбора координатной системы.

Рисунок показывает пример построения нелинейного конуса в цилиндрической системе координат (рис. 4.20). Для задания такой системы координат используется параметр coords = cylindrical.

При построении этой фигуры также использована цветная функциональная окраска. Кроме того, этот пример иллюстрирует вывод над рисунком титульной строки.

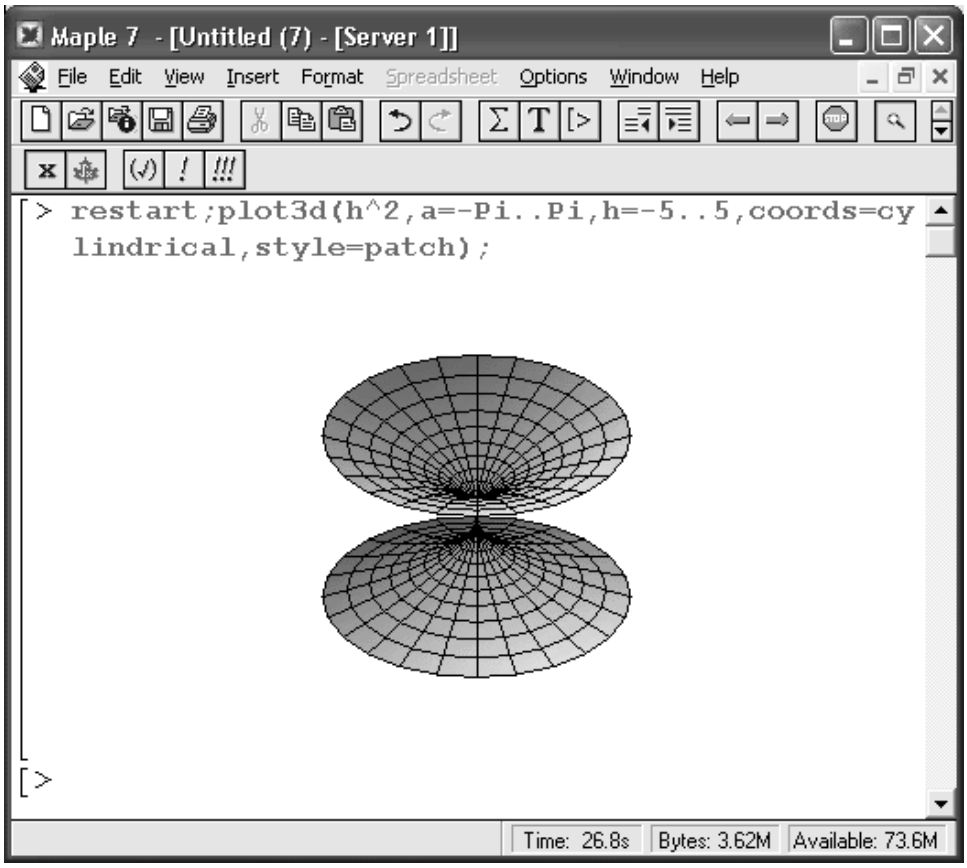


Рисунок 4.20 – Построения нелинейного конуса в цилиндрической системе координат

Приведем еще один пример построения трехмерной поверхности – в данном случае в сферической системе координат. Здесь функция задана вообще элементарно просто – в виде числа 1 (рис. 4.21). Но поскольку выбрана сферическая система координат, то в результате строится поверхность шара единичного радиуса.

О том, насколько необычным может быть график той или иной функции в различных системах координат, свидетельствует рис. 4.22:

Здесь показан график параметрически заданной функции от одной координаты t = sin(t3), построенный в сферической системе координат. Также, рис.4.22 иллюстрирует возможность одновременного наблюдения нескольких окон. В одном окне задано построение графика, а в другом построен сам график. При построении графика в отдельном окне появляется панель форматирования графика. С ее помощью можно легко скорректировать вспомогательные параметры графика (окраску, наличие линий каркаса, ориентацию и др.).

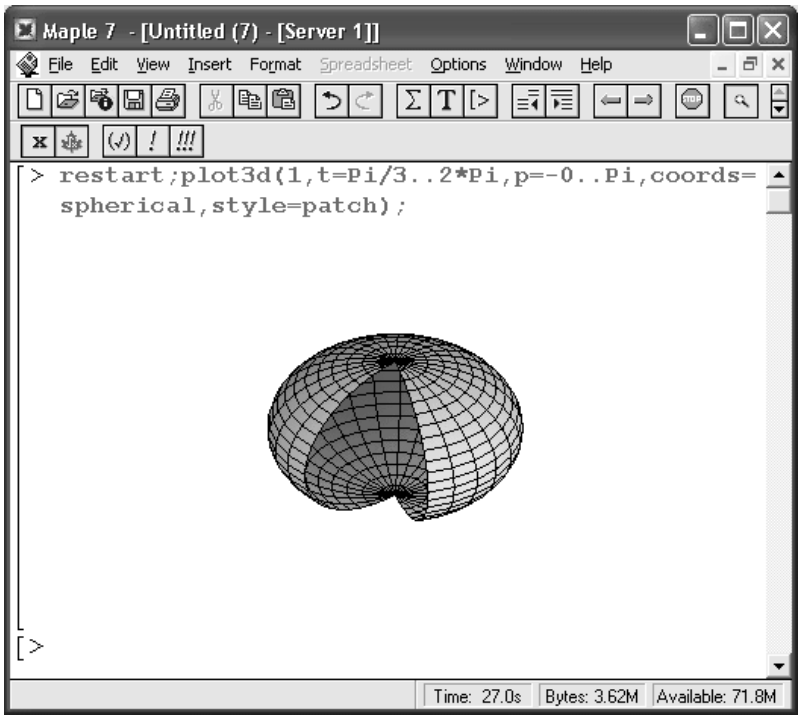


Рисунок 4.21 – Построение в сферической системе координат

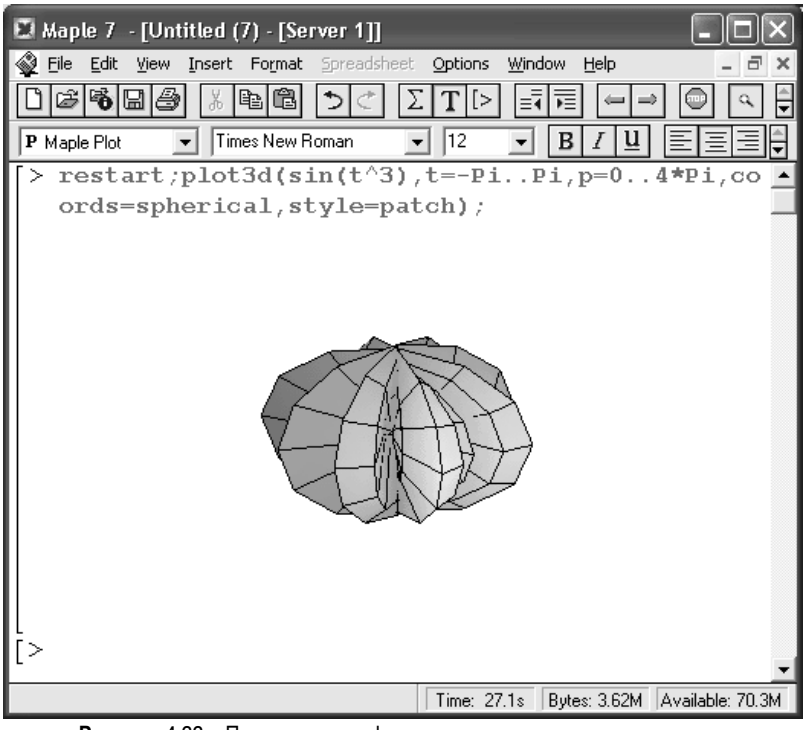


Рисунок 4.22 – Построение графика в различных системах координат

На рис. 4.23 показано построение поверхности при полном ее параметрическом задании. В этом случае поверхность задается тремя формулами, содержащимися в списке:

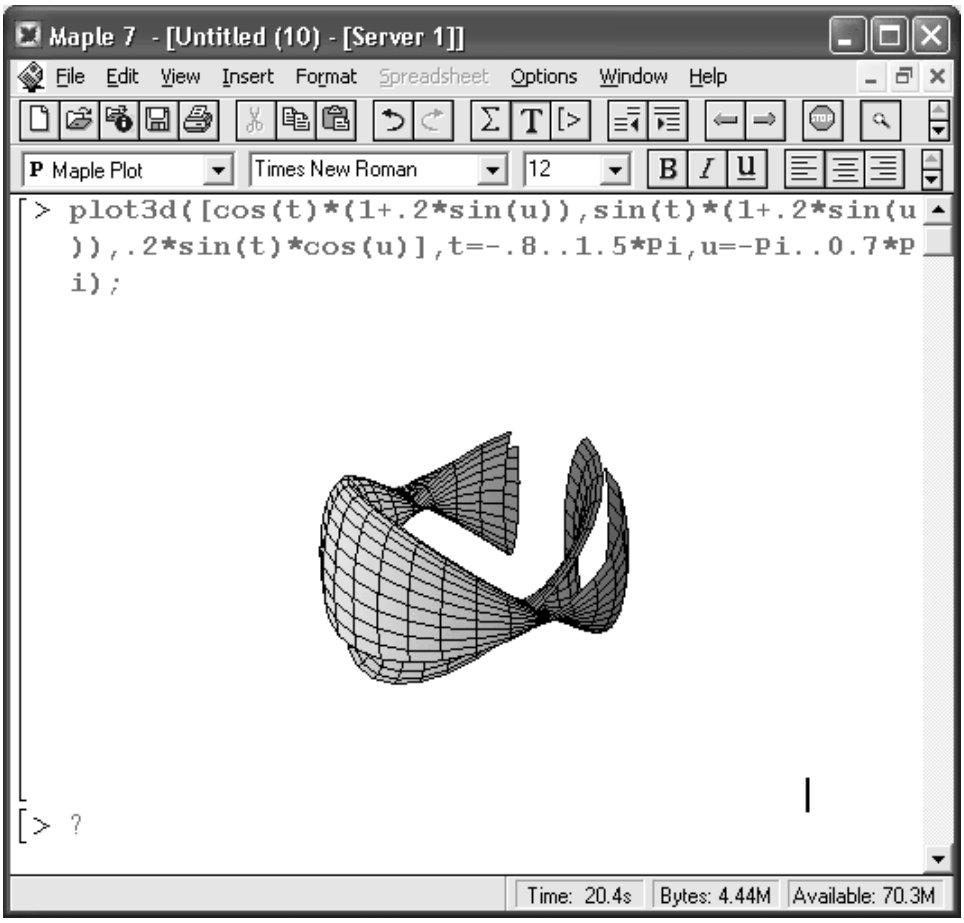


Рисунок 4.23 – Построение поверхности при полном ее параметрическом задании

В данном случае функциональная окраска задана из меню, поэтому в состав функции соответствующий параметр не введен. Обратите внимание на технику удаления частей фигуры путем задания соответствующего диапазона изменения параметров *t* и *n*.

Параметрическое задание уравнений поверхности открывает почти неисчерпаемые возможности построения занимательных и сложных фигур самого различного вида. Приведем пару построений такого рода.

На рис. 4.24 показан тор, сечение которого имеет вид сплюснутой шестиконечной звезды.

Вырез в фигуре дает прекрасный обзор ее внутренней поверхности, а цветная функциональная окраска и линии сетки, построенные с применением алгоритма удаления невидимых линий, дают весьма реалистичный вид фигуры. Замените параметр scaling = unconstrained на scaling = constrained, и вы получите тор с неискаженным сечением.

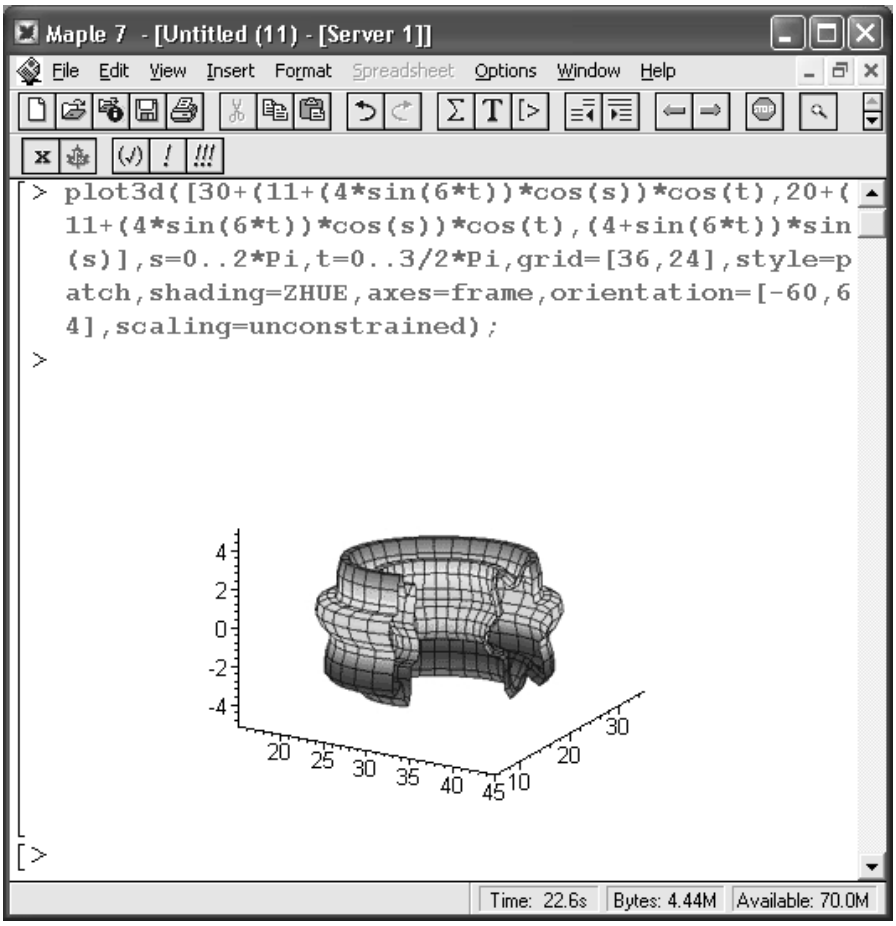


Рисунок 4.24 – Построение тора

На следующем рис. 4.25 показан еще один тор — круглого сечения, но сверху и снизу имеет вид пятиконечной звезды:

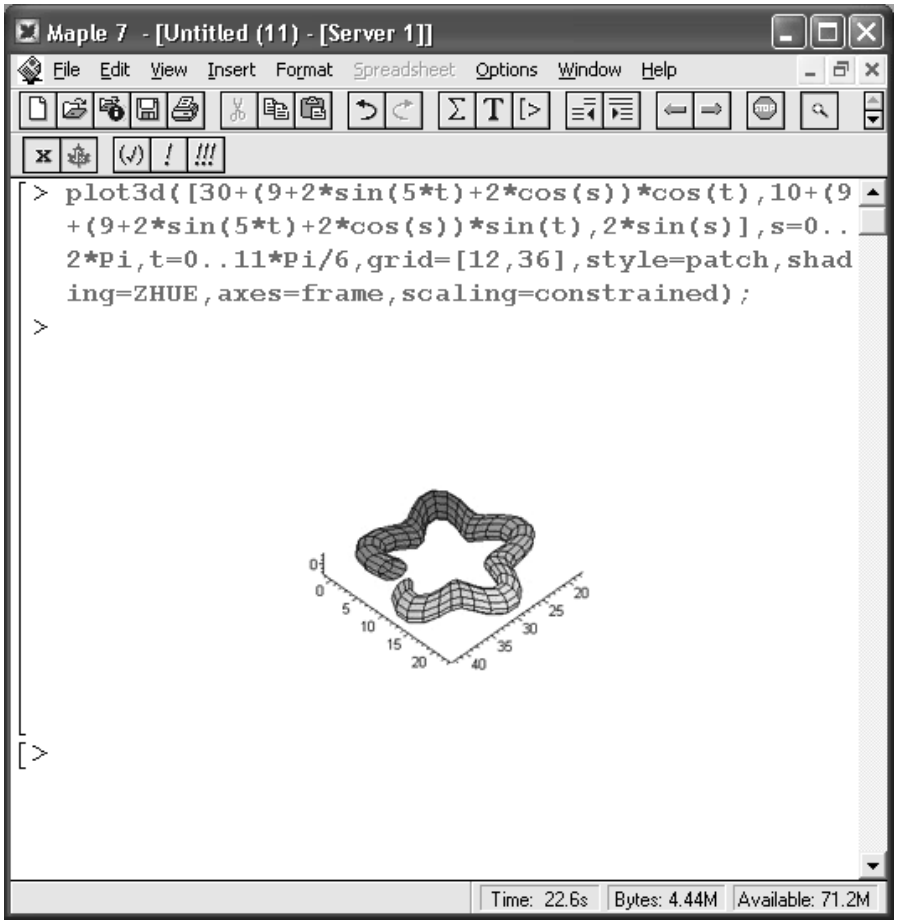


Рисунок 4.25 – Тор круглого сечения

*4.2.3. **Понятие о графических структурах***

Функции PLOT и PLOT3D (с именами, набранными большими буквами) позволяют создавать графические структуры, содержащие ряд графических объектов si, s2, s3 и т. д. Каждый объект может представлять собой точку или фигуру, полигон, надпись и т. д., позиционированную с высокой точностью в заданной системе координат. Координатные оси также относятся к графическим объектам. Важно отметить, что функции PLOT и PLOT3D одновременно являются данными, описывающими графики. Их можно записывать в виде файлов и (после открытия файлов) представлять в виде графиков. Особые свойства этих функций подчеркиваются их записью прописными буквами.

Графическая структура двумерной графики задается в виде:

$$PLOT(s1, s2, s3....,0), \qquad (4.8)$$

где: si, s2, s3... – графические объекты (или элементарные структуры – примитивы), о – общие для структуры параметры.

Основными объектами являются:

POINTS([xl,yl],[x2,y2],...[xn,yn]) – построение точек, заданных их координатами;

CURVES([[xll,yll],...[xln,yln]],[[x21,y21],...[x2п,y2п]],...[[xml,yml]....[xmn.ymn]]) – построение кривых по точкам;

POLYGONS([[xll,yll]....[xln,yln]],[x2l.y2ll]....[x21n,y21n]]....[[xml,yml]....[xmn.ynm]]) – построение замкнутой области-полигона (многоугольника, так как последняя точка должна совпадать с первой);

TEXT([x, y], 'string', horizontal .vertical) – вывод текстовой надписи 'string4, позиционированной в точке с координатами [*x; y*], с горизонтальной или вертикальной ориентацией. Параметр horizontal может иметь значение ALIGNLEFT или ALIGNRIGHT, указывающие, в какую сторону (влево или вправо) идет надпись. Аналогично параметр vertical может иметь значение ALIGNABOVE или ALIGNBELOW, указывающее в каком направлении (вверх или вниз) идет надпись.

При задании графических объектов (структур) si, s2, s3 и т. д. можно использовать описанные выше параметры и параметры, например, для задания стиля построения – STYLE (POINT, LINE/ PATCH, PATCHNOGRID); толщины линий – THICKNESS (кроме координатных осей); символа, которым строятся точки кривых – SYMBOL (BOX, CROSS, CIRCLE, POINT, DIAMOND и DEFAULT); стиля линий – LINESTYLE; цвета – COLOR (например, COLOR(HUE.O) для закраски непрерывной области), типа шрифта – FONT; вывода титульной надписи – TITLE (string); имени объекта – NAME (string); стиля координатных осей – AXESSTYLE (BOX, FRAME, NORMAL, NONE или DEFAULT) и т. д.

Следует отметить, что параметры в графических структурах задаются несколько иначе – с помощью круглых скобок. Например, для задания шрифта TIMES ROMAN с размером символов 16 пунктов надо записать FONT(TIMES,ROMAN, 16), для задания стиля координатных осей в виде прямоугольника – AXESSTYLE (BOX) и т. д.

На рис. 4.26 показан пример графических построений при использовании основных структур двухмерной графики.

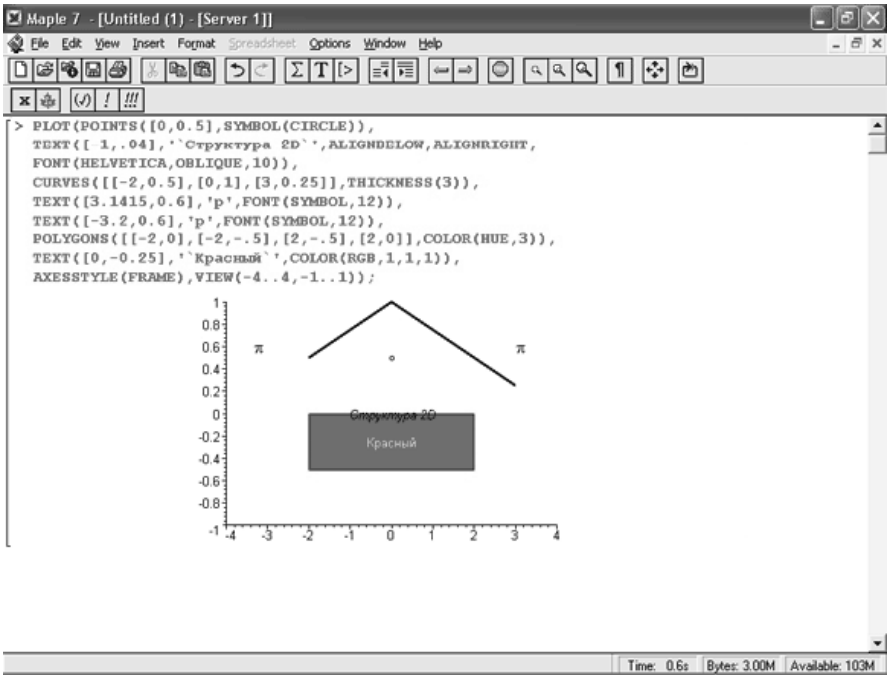


Рисунок 4.26 – Графические построения при использовании основных структур двухмерной графики

Как видно из этого примера, графическая двумерная структура позволяет задавать практически

любые двухмерные графики и текстовые надписи в пределах одного рисунка.

Графические структуры трехмерной графики строятся функцией PLOT3D:

PLOT3D(s1,s2,s3.....o). (4.9)

В качестве элементарных графических структур можно использовать уже описанные выше объекты POINTS, CURVES, POLYGONS и TEXT – разумеется, с добавлением в списки параметров третьей координаты.

Кроме того, могут использоваться некоторые специальные трехмерные структуры. Одна из них – структура GRID:

GRID(a..b,c..d.listlist) – задание поверхности над участком координатной плоскости, ограниченной отрезками [a, b] и [c, d], по данным, заданным переменной – списком listlist: = [[z11,...z1n], [z21,... z2n],...[z1m...zmn]] с размерностью nm. Заметим, что эта переменная задает координату z для равноотстоящих точек поверхности.

На рис. 4.27 показан пример создания структуры трехмерной графики на базе GRID:

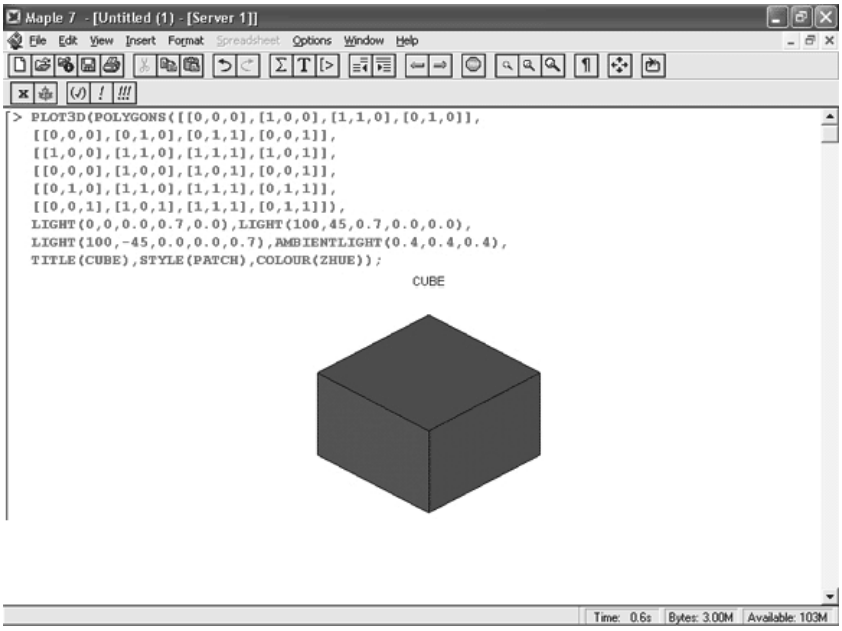


Рисунок 4.27 – Создание структуры трехмерной графики на базе GRID

Изображение представляет собой линии, соединяющие заданные точки.

Еще один тип трехмерной графической структуры – это MESH:

MESH(listlist) – задание трехмерной поверхности по данным списочной переменной list! 1st, содержащей полные координаты всех точек поверхности (рис. 4.28) (возможно задание последней при неравномерной сетке).

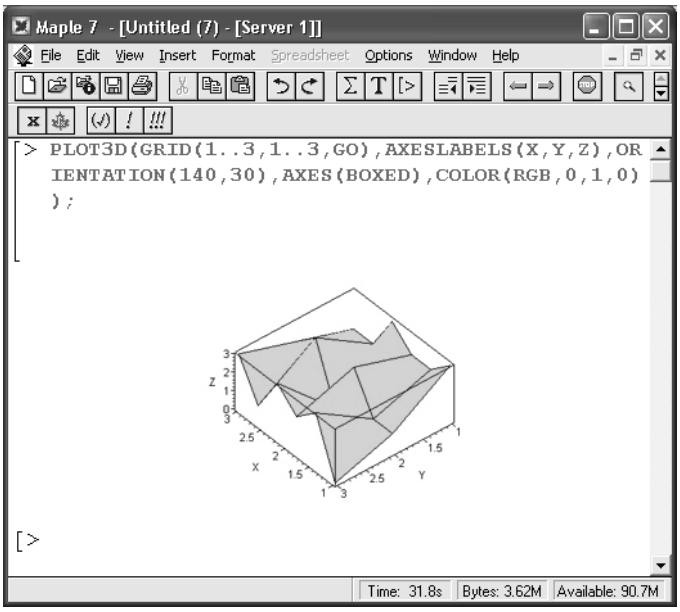


Рисунок 4.28 – Создание структуры трехмерной графики на базе MESH

Обычная форма задания этой структуры следующая:

MESH([[[x11,y11,z11]....[x1n,y1n,z1n]]. [[x21,y21,z21]....[x2n,y2n,z2n]]. ...[[xm1,ym1,zm1]... [xmn,ymn,zmn]]])

Пример задания такой структуры представлен на рис. 4.29:

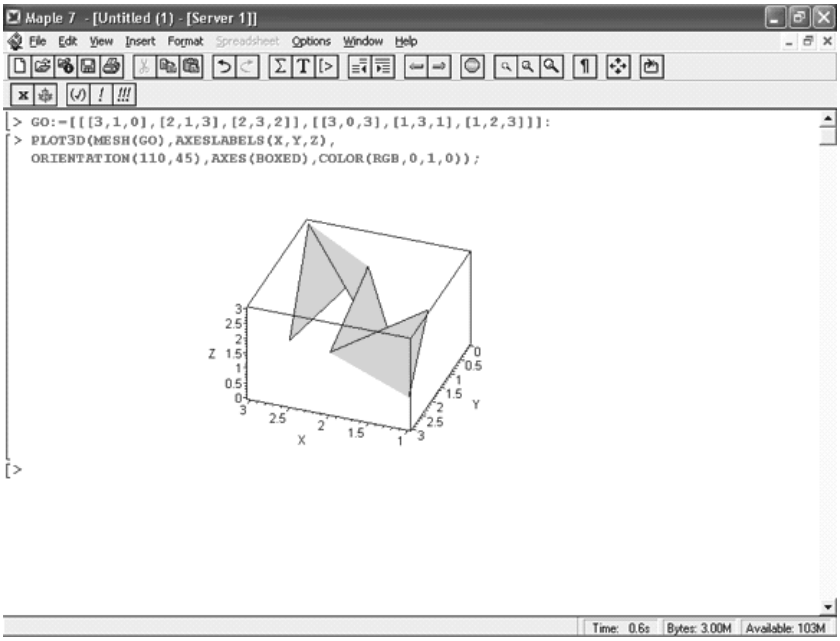


Рисунок 4.29 – Создание структуры трехмерной графики на базе MESH

4.3. **Конструирование уравнения с использованием пакета** Derive

Derive — это приложение, предназначенное для любого пользователя — студента, преподавателя или профессионала, кому нужно решать любой тип задач, связанных с математикой. Программа может делать сложные математические и алгебраические упражнения, быстро и безошибочно. Она работает с матрицами и векторами при помощи очень простого интерфейса. Она также строит любой тип графика или представления. Задачи в области арифметики, алгебры, тригонометрии, исчислений, линейной алгебры и исчисления высказываний могут быть решены одним щелчком мыши. Программа строит двухмерные и трехмерные графики математических выражений, используя разнообразные системы координат. Она обрабатывает алгебраические переменные, выражения, уравнения, функции, векторы, матрицы и логические выражения так же, как научный калькулятор обрабатывает цифры. *Derive* — одна из наиболее используемых программ в сфере математики и инженерии.

Управление программой осуществляется через систему вложенных меню, находящуюся в нижней части экрана. Переключение между пунктами меню: вперед - *пробел* или *Tab*, назад *Shift-Tab*. Быстрый выбор пункта — заглавная буква в его названии. Возврат на предыдущий уровень — *Esc*. Под меню находятся: строка сообщений и строка статуса. По названиям не сложно догадаться об их назначении.

Author — ввод нового выражения. Конец ввода — Enter или Ctrl-Enter, если хотите сразу его упростить. Кнопки F3 и F4 позволяют скопировать в строку ввода выделенное выражение (F4 — заключает его в скобки).

Build — построение выражения из уже введенных.

Calculus — позволяет произвести действия с выражением, содержит подпункты:

- Differentiate — дифференцировать;
- Integrate — интегрировать;
- Limit — предел;
- Product — произведение ряда;
- Sum — сумма ряда;
- Taylor — ряд Тейлора.

Declare — позволяет объявлять:

- Function — функции;
- Variable — переменные;
- Matrix — матрицы;
- vectoR — векторы.

Expand — раскрывает скобки.

Factor — преобразование выражения в форму:

- Trivial — простейшую;
- Squarefree — избавиться от квадратного корня;
- Rational — рациональную;
- raDical — разложить на множители;
- Complex — комплексную.

Применительно к числовому выражению Factor осуществляет разложение на множители.

Help — продемонстрирует Вам в удобной форме содержимое файла DERIVE.HLP.

Jump — переход на выражение по его номеру.

soLve — решить уравнение, неравенство, систему уравнений.

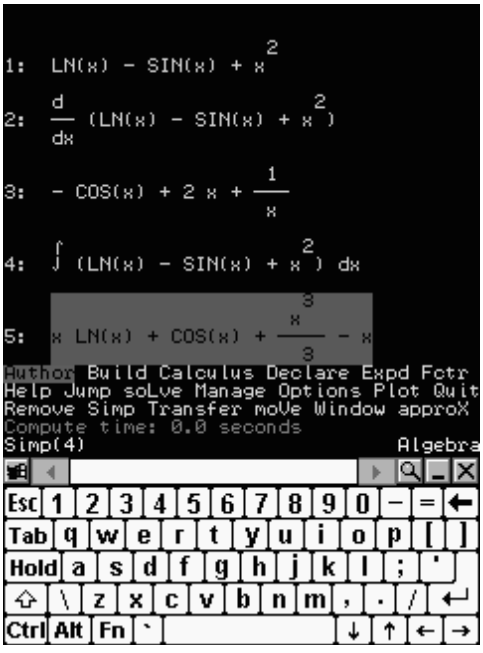


Рисунок 4.29 — Пример дифференцирования и интегрирования

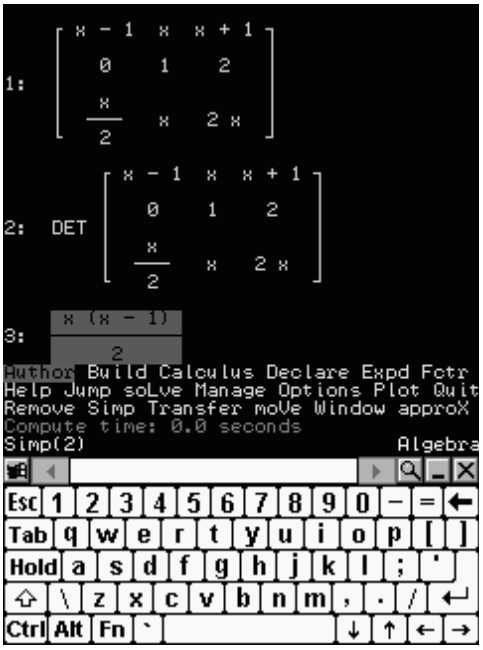


Рисунок 4.30 — Пример нахождения детерминанта матрицы

Manage — управление логикой обработки выражений. Содержит подразделы:

- Branch — определяет, является ли выражение под корнем комплексной или действительной величиной;
- Exponential — задает логику обработки выражений с показательной функцией;
- Logarithm — аналогично с логарифмами;
- Ordering — порядок следования переменных;
- Substitute — подстановка переменных;
- Trigonometry — задает логику обработки выражений с тригонометрическими функциями.

Options — разнообразные опции, рассмотрим главные:

Display — установка режима дисплея. PocketDOS может работать как с портретной, так и с альбомной ориентацией экрана. Портретная ориентация имеет преимущества в более удобной работе с экранной клавиатурой, и естественной ориентацией КПК в руке. Альбомная позволяет отображать больший объем информации без прокрутки изображения, а при установке графического режима экрана строить двух-,трехмерные графики функций. Необходимо учесть, что если у Вас русифицирована PocketDOS (присутствует файл PocketDOS.fnt), то в текстовом режиме специальные символы (знак интеграла, греческий алфавит и т.д.) будут отображаться неверно.

Для портретной ориентации выбираем:
Mode=Text,
Resolution=Medium,
Text=Large,
Set=Extendet,
Adapter=CGA.

Для альбомной ориентации выбираем:
Mode=Text,
Resolution=High,
Text=Large,
Set=Extendet,
Adapter=CGA.

Для построения графиков функций (в альбомной ориентации) выбираем:
Mode=Graphics,
Resolution=Medium,
Text=Large,
Set=Extendet,
Adapter=CGA.

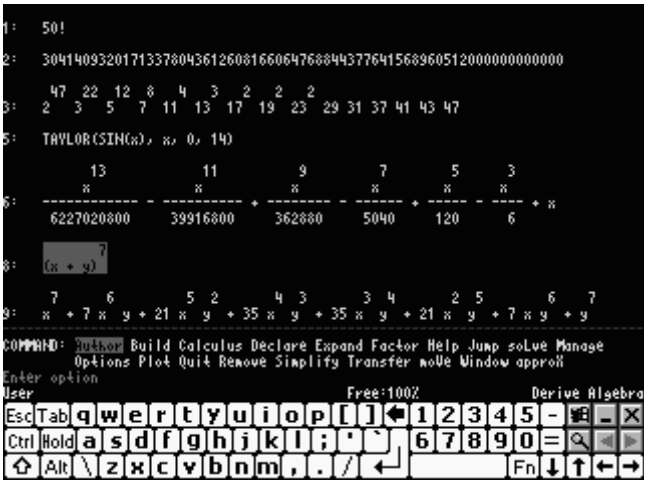


Рисунок 4.31 — Пример работы в альбомной ориентации экрана

Notation — стиль записи чисел;

Precision — количество знаков в числах. Введите в это поле 40 и сможете узнать, что число пи равно не 3.14, а *3.1415926535897932384626433832795028841 97*.

Radix — основание вводимых/выводимых чисел.

Plot — построение графиков функций.

Quit — выход из Derive, если у Вас остались не сохраненные на диске выражения, то программа запросит подтверждение операции.

Remove — удалить текущее (по умолчанию) или несколько выражений из заданного диапазона.

Simplify — упростить выражение. Результат работы этой функции сильно зависит от установок в пунктах Options и Manage.

Transfer — операции с диском:

- Load
- Derive — загрузить выражения Derive из MTH-файл;
- State — загрузка установок Derive из файла (по умолчанию DERIVE.INI);
- Utility — загрузить файл библиотеки внешних расширений.
- Save
- Derive — записать выражения Derive в MTH-файл;
- Basic, C, Fortran, Pascal — сохранение выражений в форматах Бейсик, Си, Фортран и Паскаль соответственно;
- Options — диапазон сохраняемых выражений и длина строки;
- State — сохранение установок *Derive* в файле
- Print — печать.
- Window — операции с окнами.
- approX — упростить и показать результат в десятичной нотации.

1. В качестве несложного примера найдем корни квадратного уравнения. Выбираем Author, вводим: *6 x x — 13 x + 6 = 0* Выбираем soLve получаем:

$$x = \frac{2}{3} \qquad x = \frac{3}{2}$$

2. Уточним, по каким формулам *Derive* это посчитал. Выбираем Author, вводим: *a x x + b x + c = 0* Выбираем soLve, получаем:

$$x = \frac{SQRT(b^2 - 4\,a\,c) - b}{2\,a}$$

$$x = -\frac{SQRT(b^2 - 4\,a\,c) + b}{2\,a}$$

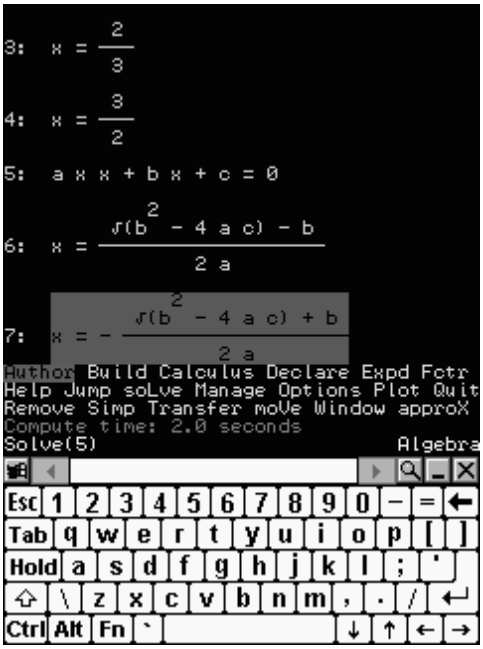


Рисунок 4.32 — Пример решения квадратного уравнения

3. Рассмотрим построение графиков функций в *Derive*.

Настроим PocketDOS на альбомный режим. Для этого в меню PocketDOS выбираем:

Setting->Display->Orientation->Landscape(rotated CW)

Выходим из PocketDOS и снова ее запускаем. Запускаем Derive. Настраиваем его на работу в графическом режиме:

Options->Display

Mode: Graphics

Resolution: Medium

4. Построение графика функции двух переменных. Такие графики *Derive* строит в виде проволочного каркаса, удаляя при этом невидимые линии.

Author

x x + z z

Plot PLOT: Overlay Grids

x: 20

y: 20

Plot

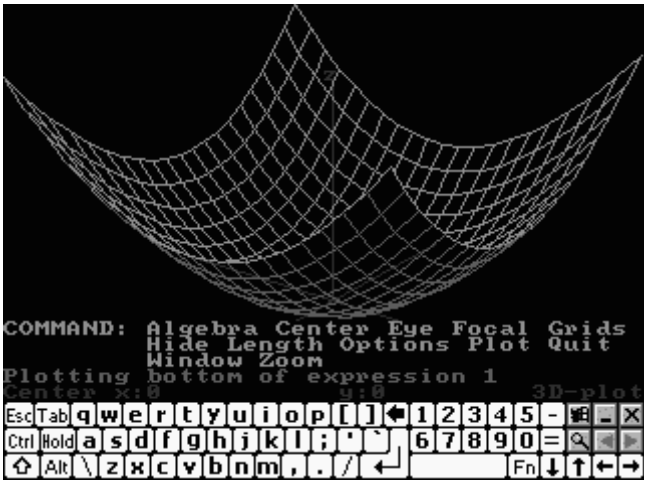


Рисунок 4.33 — Пример построения графика функции двух переменных

4.4 **Конструирование объектов с использованием пакета** MATLAB

MATLAB (сокращение от англ. *«MatrixLaboratory»*, в русской транскрипции произносится, как Матла́б) — пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете. *MATLAB* используют более 1 000 000 инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS, Solaris (начиная с версии R2010b поддержка Solaris прекращена) и Microsoft Windows.

MATLAB как язык программирования был разработан Кливом Моулером (англ. *Clev eMoler*) в конце 1970-х годов, когда он был деканом факультета компьютерных наук в Университете Нью-Мексико. Целью разработки служила задача дать студентам факультета возможность использования программных библиотек Linpack и EISPACK без необходимости изучения Фортрана. Вскоре новый язык распространился среди других университетов и был с большим интересом встречен учёными, работающими в области прикладной математики. До сих пор в Интернете можно найти версию 1982 года, написанную на Фортране, распространяемую с открытым исходным кодом. Инженер Джон Литтл (англ. *John N. (Jack) Little*) познакомился с этим языком во время визита Клива Моулера в Стэндфордский университет в 1983 году. Поняв, что новый язык обладает большим коммерческим потенциалом, он объединился с Кливом Моулером и Стивом Бангертом (англ. *Steve Bangert*). Совместными усилиями они переписали MATLAB на C и основали в 1984 году компанию The MathWorks для дальнейшего развития. Эти переписанные на C библиотеки долгое время были известны под именем JACKPAC. Первоначально *MATLAB* предназначался для проектирования систем управления (основная специальность Джона Литтла), но быстро завоевал популярность во многих других научных и инженерных областях. Он также широко использовался и в образовании, в частности, для преподавания линейной алгебры и численных методов.

*4.4.1. **Описание языка** MATLAB*

Язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

Программы, написанные на *MATLAB*, бывают двух типов — функции и скрипты. Функции имеют входные и выходные аргументы, а также собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных. Скрипты же используют общее рабочее пространство. Как скрипты, так и функции не компилируются в машинный код и сохраняются в виде текстовых файлов. Существует также возможность сохранять так называемые *pre-parsed* программы — функции и скрипты, преобразованные в вид, удобный для машинного исполнения. В общем случае такие программы выполняются быстрее обычных, особенно если функция содержит команды построения графиков.

Основной особенностью языка *MATLAB* являются его широкие возможности по работе с матрицами, которые создатели языка выразили в лозунге «думай векторно» (англ. *Think vectorized*). Пример кода, являющегося частью функции *magic.m*, генерирующего магический квадрат *M* для нечётных значений размера стороны *n*:

```
[J,I] = meshgrid(1:n);

A = mod(I+J — (n+3)/2,n);

B = mod(I+2*J — 2,n);

M = n*A + B + 1.
```

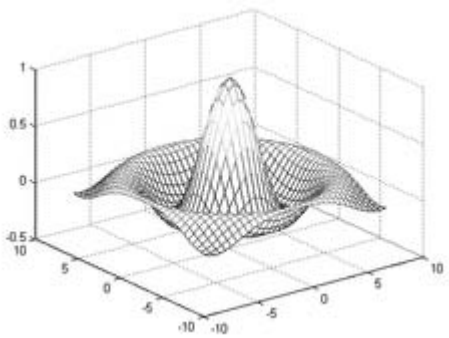
Пример кода, загружающего одномерный массив *A* значениями массива *B* в обратном порядке (только если вектор *A* определен, и число его элементов совпадает с числом элементов вектора *B*):

```
A(1:end) = B(end:—1:1);
```

Программа *MATLAB* может создавать трехмерную графику с помощью функций surf, plot3 или mesh.

```
[X,Y] = meshgrid(—8:.5:8);
R = sqrt(X.^2 + Y.^2);
Z = sin(R)./R;
Z(R==0) = 1;
mesh(X,Y,Z);
```

Этот код создаст каркасный 3D график sinc-функции $\frac{\sin R}{R}$.



Пример графика sinc-функции, нарисованного с помощью *MATLAB*

MATLAB предоставляет пользователю большое количество (несколько сотен) функций для анализа данных, покрывающие практически все области математики, в частности:

- Матрицы и линейная алгебра — алгебра матриц, линейные уравнения, собственные значения и вектора, сингулярности, факторизация матриц и другие.
- Многочлены и интерполяция — корни многочленов, операции над многочленами и их дифференцирование, интерполяция и экстраполяция кривых и другие.
- Математическая статистика и анализ данных — статистические функции, статистическая регрессия, цифровая фильтрация, быстрое преобразование Фурье и другие.

- Обработка данных — набор специальных функций, включая построение графиков, оптимизацию, поиск нулей, численное интегрирование (в квадратурах) и другие.
- Дифференциальные уравнения — решение дифференциальных и дифференциально-алгебраических уравнений, дифференциальных уравнений с запаздыванием, уравнений с ограничениями, уравнений в частных производных и др.
- Разреженные матрицы — специальный класс данных пакета *MATLAB*, использующийся в специализированных приложениях.
- Целочисленная арифметика — выполнение операций целочисленной арифметики в среде *MATLAB*.

MATLAB предоставляет также удобные средства для разработки алгоритмов, включая высокоуровневые с использованием концепций объектно-ориентированного программирования. В нём имеются все необходимые средства интегрированной среды разработки, включая отладчик и профайлер. Функции для работы с целыми типами данных облегчают создание алгоритмов для микроконтроллеров и других приложений, где это необходимо.

В составе пакета *MATLAB* имеется большое количество функций для построения графиков, в том числе трёхмерных, визуального анализа данных и создания анимированных роликов.

Встроенная среда разработки позволяет создавать графические интерфейсы пользователя с различными элементами управления, такими как кнопки, поля ввода и другими.

Программы *MATLAB*, как консольные, так и с графическим интерфейсом пользователя, могут быть собраны с помощью компоненты *MATLAB Compiler* в независимые от *MATLAB* исполняемые приложения или динамические библиотеки, для запуска которых на других компьютерах, однако, требуется установка свободно распространяемой среды *MATLAB Compiler Runtime* (MCR).

Пакет *MATLAB* включает различные интерфейсы для получения доступа к внешним подпрограммам, написанным на других языках программирования, данным, клиентам и серверам, общающимся по технологии Component Object Model или Dynamic Data Exchange, а также периферийным устройствам, которые взаимодействуют напрямую с *MATLAB*. Многие из них известны под названием *MATLAB API*.

Пакет *MATLAB* предоставляет доступ к функциям, позволяющим создавать, манипулировать и удалять COM-объекты (как клиенты, так и серверы). Поддерживается также технология ActiveX. Все COM-объекты принадлежат к специальному COM-классу пакета *MATLAB*. Все программы, имеющие функции контроллера автоматизации (англ. *Automation controller*) могут иметь доступ к *MATLAB* как к серверу автоматизации (англ. *Automation server*).

Пакет *MATLAB* в Microsoft Windows предоставляет доступ к программной платформе .NET Framework. Имеется возможность загружать .NET сборки (Assemblies) и работать с объектами .NET классов из среды MATLAB. В версии *MATLAB* 7.11 (R2010b) поддерживается .NET Framework версий 2.0, 3.0, 3.5 и 4.0.

Пакет *MATLAB* содержит функции, которые позволяют ему получать доступ к другим приложениям среды Windows, равно как и этим приложениям получать доступ к данным *MATLAB*, посредством технологии динамического обмена данными (DDE). Каждое приложение, которое может быть DDE-сервером, имеет своё уникальное идентификационное имя. Для *MATLAB* это имя — *Matlab*.

В *MATLAB* существует возможность вызывать методы веб-сервисов. Специальная функция создаёт класс, основываясь на методах API веб-сервиса.

MATLAB взаимодействует с клиентом веб-сервиса с помощью принятия от него посылок, их обработки и посылок ответа. Поддерживаются следующие технологии: Simple Object Access Protocol (SOAP) и Web Services Description Language (WSDL).

Интерфейс для последовательного порта пакета *MATLAB* обеспечивает прямой доступ к периферийным устройствам, таким как модемы, принтеры и научное оборудование, подключающееся к компьютеру через последовательный порт (COM-порт). Интерфейс работает путём создания объекта специального класса для последовательного порта. Имеющиеся методы этого класса позволяют считывать и записывать данные в последовательный порт, использовать события и обработчики событий, а также записывать информацию на диск компьютера в режиме реального времени. Это бывает необходимо при проведении экспериментов, симуляции систем реального времени и для других приложений.

Пакет *MATLAB* включает интерфейс взаимодействия с внешними приложениями, написанными на языках C и Фортран. Осуществляется это взаимодействие через MEX-файлы. Существует возможность вызова подпрограмм, написанных на C или Фортране из *MATLAB*, как будто это встроенные функции пакета. MEX-файлы представляют собой динамически подключаемые библиотеки, которые могут быть загружены и исполнены интерпретатором, встроенным в *MATLAB*. MEX-процедуры имеют также возможность вызывать встроенные команды *MATLAB*.

Интерфейс *MATLAB*, относящийся к общим DLL позволяет вызывать функции, находящиеся в обычных динамически подключаемых библиотеках, прямо из *MATLAB*. Эти функции должны иметь C-интерфейс.

4.4.2. ***Наборы инструментов***

Для *MATLAB* имеется возможность создавать специальные наборы инструментов (англ. *toolbox*), расширяющие его функциональность. Наборы инструментов представляют собой коллекции функций, написанных на языке MATLAB для решения определённого класса задач. Компания Mathworks поставляет наборы инструментов, которые используются во многих областях, включая ниже следующие:

- Цифровая обработка сигналов, изображений и данных: *DSPToolbox*, *Image Processing Toolbox*, *Wavelet Toolbox*, *Communication Toolbox*, *Filter Design Toolbox* — наборы функций, позволяющих решать широкий спектр задач обработки сигналов, изображений, проектирования цифровых фильтров и систем связи.
- Системы управления: *Control Systems Toolbox*, *μ-Analysisand Synthesis Toolbox*, *Robust Control Toolbox*, *System Identification Toolbox*, *LMI Control Toolbox*, *Model Predictive Control Toolbox*, *Model-Based Calibration Toolbox* — наборы функций, облегчающих анализ и синтез динамических систем, проектирование, моделирование и идентификацию систем управления, включая современные алгоритмы управления, такие как робастное управление, H ∞-управление, ЛМН-синтез, μ-синтез и другие.
- Финансовый анализ: *GARCH Toolbox*, *Fixed-Income Toolbox*, *Financial Time Series Toolbox*, *Financial Derivatives Toolbox*, *Financial Toolbox*, *Datafeed Toolbox* — наборы функций, позволяющие быстро и эффективно собирать, обрабатывать и передавать различную финансовую информацию.
- Анализ и синтез географических карт, включая трёхмерные: *Mapping Toolbox*.
- Сбор и анализ экспериментальных данных: *Data Acquisition Toolbox*, *Image Acquisition Toolbox*, *Instrument Control Toolbox*, *Linkfor Code Composer Studio* — наборы функций, позволяющих сохранять и обрабатывать данные, полученные в ходе экспериментов, в том числе в реальном времени. Поддерживается широкий спектр научного и инженерного измерительного оборудования.

- Визуализация и представление данных: *Virtual Reality Toolbox* — позволяет создавать интерактивные миры и визуализировать научную информацию с помощью технологий виртуальной реальности и языка VRML.
- Средства разработки: *MATLAB Builderfor COM*, *MATLAB Builder for Excel*, *MATLAB Builder for NET*, *MATLAB Compiler*, *Filter Design HDL Coder* — наборы функций, позволяющих создавать независимые приложения из среды MATLAB.
- Взаимодействие с внешними программными продуктами: *MATLAB Report Generator*, *Excel Link*, *Database Toolbox*, *MATLAB Web Server*, *Linkfor Model Sim* — наборы функций, позволяющие сохранять данные в различных видах таким образом, чтобы другие программы могли с ними работать.
- Базы данных: *Database Toolbox* — инструменты работы с базами данных.
- Научные и математические пакеты: *Bioin formatics Toolbox*, *Curve Fitting Toolbox*, *Fixed-Point Toolbox*, *Fuzzy Logic Toolbox*, *Genetic Algorithmand Direct Search Toolbox*, *OPC Toolbox*, *Optimization Toolbox*, *Partial Differential Equation Toolbox*, *Spline Toolbox*, *Statistic Toolbox*, *RF Toolbox* — наборы специализированных математических функций, позволяющие решать широкий спектр научных и инженерных задач, включая разработку генетических алгоритмов, решения задач в частных производных, целочисленные проблемы, оптимизацию систем и другие.
- Нейронные сети: *Neural Network Toolbox* — инструменты для синтеза и анализа нейронных сетей.
- Нечёткая логика: *Fuzzy Logic Toolbox* — инструменты для построения и анализа нечётких множеств.
- Символьные вычисления: *Symbolic Math Toolbox* — инструменты для символьных вычислений с возможностью взаимодействия с символьным процессором программы Maple.
- Помимо вышеперечисленных, существуют тысячи других наборов инструментов для MATLAB, написанных другими компаниями и энтузиастами.

*4.4.3. **Пример моделирования в в программе** Matlab*

1. Построение графика функций одной переменной

```
>> x=0:0.1:10;

>> plot(sin(x))
```

Рассмотрим вначале простейший пример — построение графика синусоиды. Следует помнить, что *MATLAB* строит графики функций по ряду точек, соединяя их отрезками прямых, т. е. осуществляя линейную интерполяцию функции в интервале между смежными точками. Зададим интервал изменения аргумента от 0 до 10 с шагом 0.1. Для построения графика достаточно вначале задать вектор , а затем использовать команду построения графиков. Результат построения показан на рис. 4.33.

Вектор x задает интервал изменения независимой переменной от 0 до 10 с шагом 0.1. plot строит не истинный график функции sin(x) (рис. 4.33), а лишь заданное числом элементов вектора x число точек. Эти точки затем просто соединяются отрезками прямых, т. е. осуществляется кусочно-линейная интерполяция данных графика. При 100 точках, полученная кривая воспринимается глазом как вполне плавная, но при 10 - 20 точках она будет выглядеть состоящей из отрезков прямых.

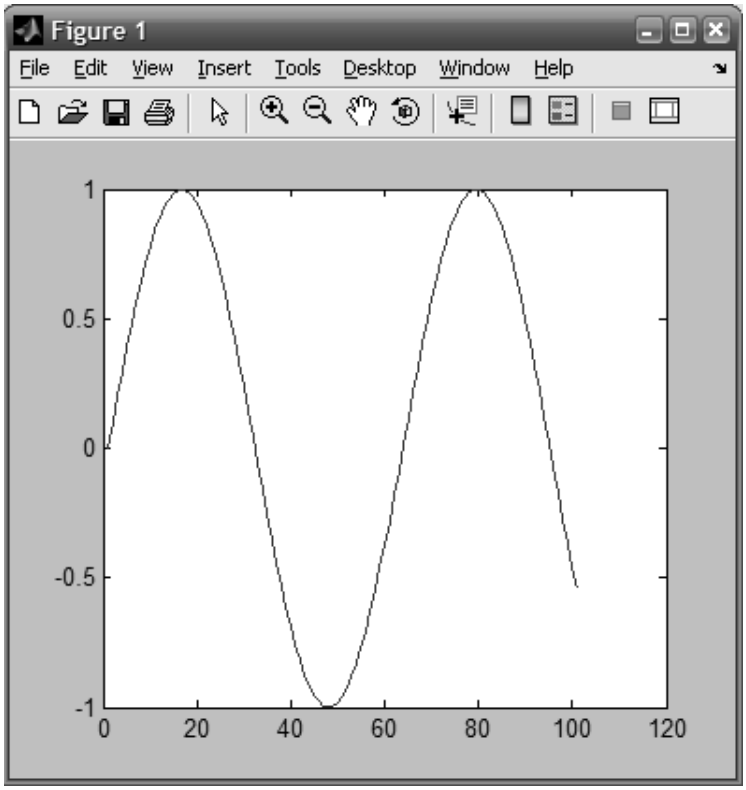


Рисунок 4.33 – График функции sin(*x*))

2 Пойдем дальше и попытаемся построить графики сразу трех функций: sin(x), cos(x) и sin(x)

yl=sin(x); y2=cos(x); y3=sin(x)/x;

Отметим, что переменные yl, y2, y3 в результате выполнения данных команд являются не функциями, а векторами, имеющими ту же размерность, что и вектор. Поэтому эти переменные не имеют явного указания аргумента в виде *y(x).*

Теперь можно использовать одну из форм команды plot:

>> plot(xl,fl,x2,f2,x3,f3,...)

где: xl, x2, x3,... — векторы аргументов функций (в нашем случае все они равны *x*), f1, f2, f3,... — векторы значений функций, графики которых строятся в одном окне.

В нашем случае для построения графиков указанных функций мы должны записать следующее: « plot(x,yl,x,y2,x,y3)

Результат показан на рис. 4.34.

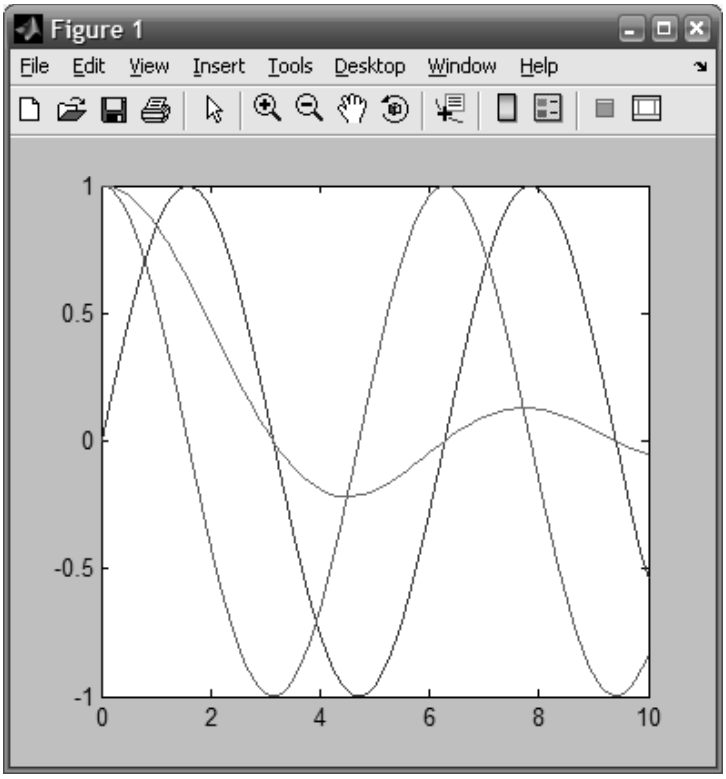


Рисунок 4.34 – График нескольких функции

3. Построение трехмерных графиков.

Для построения графика поверхности (рис. 4.35) и ее проекции в виде контурного графика на плоскость под поверхностью достаточно использовать следующие команды:

« [X,Y]=meshgrid(-5:0.1:5);

« Z=X.*sin(X+Y);

« meshc(X,Y,Z)

Трехмерные поверхности обычно описываются функцией двух переменных z *(x, y).* Специфика построения трехмерных графиков требует не просто задания ряда значений x и y, то есть векторов x и y. Она требует определения для *X* и *Y* двухмерных массивов — матриц. Для создания таких массивов служит функция meshgrid. В основном она используется совместно с функциями построения графиков трехмерных поверхностей. Функция meshgrid записывается в следующих формах:

[X, Y] = meshgrid(x) — аналогична [X, Y] = meshgrid(x.x);

[X, Y, Z] = meshgrid(x, y, z) — возвращает трехмерные массивы, используемые для вычисления функций трех переменных и построения трехмерных графиков;

[X, Y] = meshgrid(x, y) — преобразует область, заданную векторами x и y, в массивы X и Y, которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков. Строки выходного массива X являются копиями вектора x; а столбцы Y — копиями вектора y.

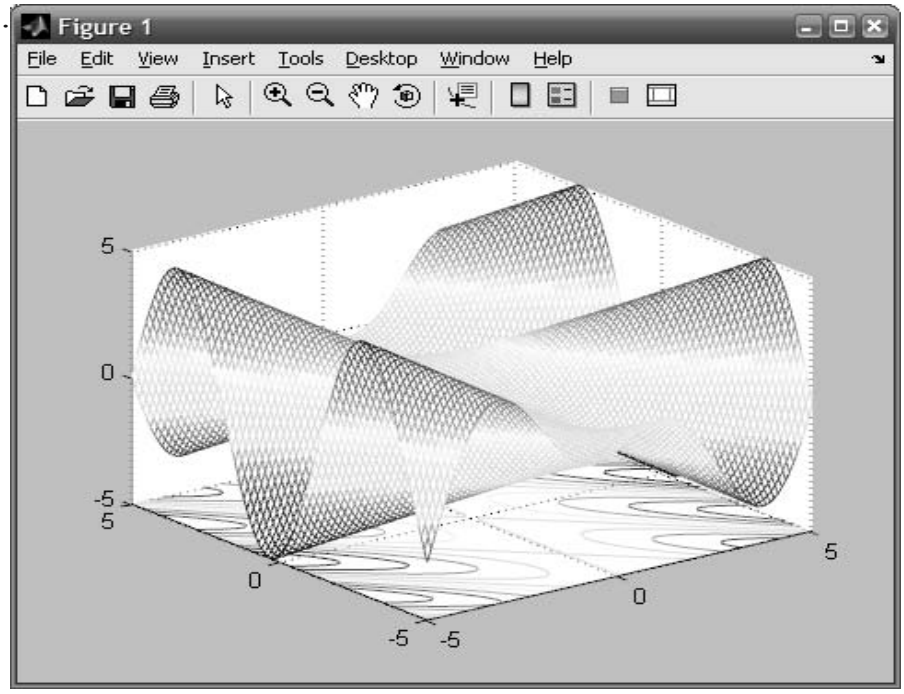


Рисунок 4.35– График поверхности

4.5. **Конструирование объектов с использованием пакета** MathCAD

Допустим, необходимо решить уравнение $(\frac{\tilde{o}}{2})^2 + \sin^2(2x) = 3\ln|x|$.

Решение.

Решение данного уравнения будем проводить в два этапа: отделение корней уравнения графически, уточнение корней уравнения.

Определим функцию *f*(*x*), равную левой части данного уравнения, когда правая равна нулю:

$$f(x) = (\frac{\tilde{o}}{2})^2 + \sin^2(2x) = 3\ln|x|$$

Зададим ранжированную переменную *x* на некотором диапазоне с мелким шагом, например:

$$x := -5, -4.9..5$$

Вставим в документ графическую область. Для этого выберем дважды пиктограмму с изображением графика сначала на панели Math (*Математика*), затем на палитре графиков Graph или выполним из главного меню последовательность команд Insert / Graph / X-Y Plot (*Вставка / График / X—Y Зависимость*).

Снизу по оси абсцисс наберем *x*, а сбоку по оси ординат введем *f*(*x*).

Для появления графика щелкнем левой клавишей мыши вне графической области.

Отформатируем график функции *f*(*x*). Для этого щелкнем правой клавишей мыши в области графика и выберем в контекстном меню команду Format (*Формат*).

Установим пересечение осей графика (Crossed − *Только оси*), добавим вспомогательные линии по координатным осям (Grid Lines − *Вспомогательные линии*).

Отменим при этом автосетку (Autogrid − *Автосетка*) и установим количество линий сетки, равное 10.

Для подтверждения внесенных изменений нажмем последовательно кнопки Apply (*Применить*) и OK.

После указанных преобразований график функции *f*(*x*) будет выглядеть следующим образом (рис.4.36):

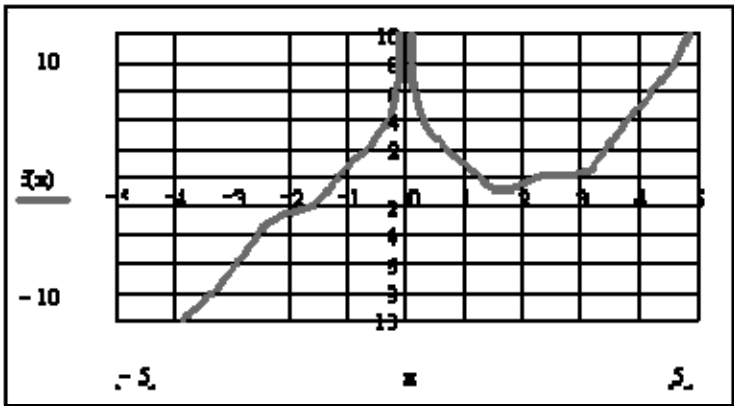


Рисунок 4.36 – График функции *f*(*x*)

Из графика функции *f*(*x*) видно, что уравнение $(\frac{\tilde{o}}{2})^2 + \sin^2(2x) = 3\ln|x|$ имеет три корня, которые приблизительно равны: $x_1 \approx -1$; $x_2 \approx 1$; $x_3 \approx 2{,}5$.

Этап отделения корней завершен.

Уточним теперь корни уравнения различными способами.

1-й способ. Присвоим начальное приближение переменной *x* и укажем точность поиска корня:

```
x :=     TOL := 0.0001
```

Уточним заданное приближение к значению корня с помощью функции *root*:

```
x1 := root (f(x), x)

x1 = – 1.1395
```

Выполним проверку, подтверждающую, что первый корень найден с заявленной точностью:

```
f(x1) := – 2.8866×10^-14
```

Начальное приближение можно не задавать при использовании в качестве аргументов *root* границ отрезка нахождения корня, например, второй корень можно уточнить:

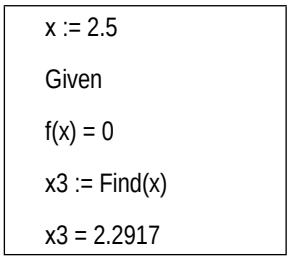
```
root (f(x), x, 1, 2) = 1.232
```

2-й способ. Присвоим начальное приближение переменой *x* для уточнения третьего корня:

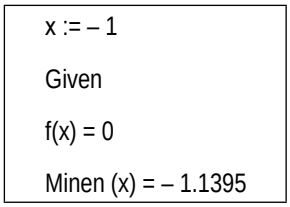
```
x := 2.5
```

Напечатаем служебное слово *Given*. Ниже наберем *f*(*x*) = 0, используя логический знак равенства с панели Boolean (*Логические*) − комбинация клавиш Ctrl + =. Еще ниже напечатаем выражение *x*3 = Find(*x*) и вывод значения третьего корня *x*3 =.

Уточнение корня уравнения с помощью блока *Given…Find* выглядит следующим образом:



3-й способ. Уточнение корня уравнения с помощью блока *Given…Minerr* осуществляется аналогично предыдущему случаю:



Примечание. Для уточнения корня в данном примере установлена точность 0,0001. Поэтому целесообразно изменить формат вывода результатов (4 знака после десятичного разделителя) в окне форматирования результатов на вкладке Number Format.

Таким образом, с помощью инструментов в MathCAD-е возможно решение уравнения с построением графика.

ВЫВОДЫ

Программа *Maple* корпорации WaterlooMapleInc. – патриарх в мире систем компьютерной математики. Эта система, снискавшая себе мировую известность и огромную популярность, является одной из лучших среди систем символьной математики, позволяющих решать математические задачи в аналитическом виде.

Не стоит забывать, что *Maple* многогранна, она может применяться для решения самых серьезных математических задач аэродинамики, теории поля, теплопроводности и диффузии, теоретической механики, с другой стороны, Maple используют для создания web-страниц – основы Интернета.

Математические пакеты, такие как *Mathcad, Maple, Mathematica*, хорошо приспособлены к проведению расчетов в естественнонаучных дисциплинах, когда модель задана в аналитической форме. Удобство варьирования параметров в сочетании с заранее определенной процедурой обработки и визуализации результатов существенно облегчает исследования. В таких многовариантных расчетах накладные расходы, связанные с написанием специальной программы на языке пакета, управляющей экспериментом, окупаются той легкостью, с которой возможно повторить все вычисления заново при внесении изменений в исходную модель. Программирование сводится к написанию относительно небольших по объему программ, состоящих в основном из макрооператоров.

С точки зрения моделирования мехатронных объектов основным и, пожалуй, единственным достоинством систем компьютерной математики является математическая прозрачность вычислений и легкость создания объектов, осуществляющих математические вычисления. К числу недостатков можно отнести отсутствие таких принципиально важных возможностей, как:

- автоматизация построения математической модели;
- компонентное моделирование с применением достаточно большого количества типовых блоков;
- быстрая модификация модели;
- создание предметно-ориентированной среды;
- оперативное изменение метода моделирования и т.д.

В результате, применение систем компьютерной математики ограничивается решением простых задач, или задач, где главное – прозрачность вычислений.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как получить передаточную функцию по линейным дифференциальным уравнениям системы?
2. С помощью каких операций (функций) строятся в MATLAB модели параллельного и последовательного соединений, системы с обратной связью?
3. Как определяются запасы устойчивости по амплитуде и по фазе? Что означают эти величины? В каких единицах они измеряются?
4. Какие возможности предоставляет модуль SISOTool?
5. Что такое:
   - корневой годограф;
   - перерегулирование;
   - время переходного процесса?
6. Что такое астатическая система? Что такое порядок астатизма?
7. Основные понятия теории динамических систем в евклидовом пространстве.

8. Устойчивые движения динамических моделей. Основные понятия теории устойчивости по Ляпунову.

9. Опишите возможности MathCAD.

10. Опишите интерфейс MathCAD.

11. Какие виды массивов существуют в MathCAD?

12. Какие инструментальные средства имеются в MathCAD для построения графиков?

13. Что такое «компьютерная математика»?

14. Классификация средств компьютерной математики?

15. Особенности систем компьютерной математики?

16. Элементы теории математического моделирования динамических объектов. Метрические и нормированные пространства.

17. Элементы теории математического моделирования динамических объектов. Операторы и функционалы в метрических пространствах.

18. Основные понятия теории динамических систем в евклидовом пространстве.

19. Устойчивые движения динамических моделей. Основные понятия теории устойчивости по Ляпунову.

20. Два базовых метода построения математических моделей. Задача идентификации.

21. Применение методов оптимизации в математическом моделировании.

22. Параметрическая оптимизация с заданием допустимой динамической области.

23. Основные элементы пакета Maple. Операции с матрицами и полиномами.

24. Элементарная графика в пакете Maple. Основы программирования на языке Maple.

25. Что такое Derive?

26. Какие опции есть в Derive?

5 СИСТЕМЫ ТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ

Моделирование технических систем – упрощенное отображение реального изделия и его описания с целью оценки соответствия его какому-либо требованию или осуществления выбора наилучшего изделия из нескольких альтернативных вариантов. Обычно используют три способа моделирования технических систем:

1. Мысленное моделирование технических систем, в ходе которого человек, изучая изделие, его проект или другое описание, интуитивно оценивает соответствие определенным требованиям или выбор наилучшего варианта.

2. Математическое моделирование технических систем связано с разработкой способов расчета и компьютерных программ для получения необходимых оценок.

3. Физическое моделирование технических систем связано с изготовлением и испытанием упрощенных физических моделей реального изделия.

Мысленное моделирование технических систем основывается на знаниях и, главное, на собственном опыте проектирования и эксплуатации данного класса технических систем и представляет собой одно из средств их моделирования (рис. 5.1). Точность мысленного моделирования зависит от личного опыта и природных способностей эксперта и для мало изученных технических систем может превосходить точность математической модели. Основные преимущества мысленного моделирования: непродолжительное время и низкая стоимость оценки. Умение осуществлять быстрое и точное мысленное моделирование является одним из необходимых качеств изобретателей и творческих личностей, которые должны его развивать и совершенствовать.

При физическом моделировании решение принимается по измеряемым параметрам, данным измерительных приборов и оборудования, способу обработки полученных результатов. С целью снижения трудоёмкости и стоимости часто изготавливают уменьшённые (в несколько раз, на порядок и более) образцы технических систем, исключая из них малозначимые детали. При изменении масштаба технические системы выбирают и обосновывают систему критериев подобия, с помощью которых выполняют перерасчет значений параметров, полученных путем измерений на уменьшенных моделях, для натуральных размеров. В различных прикладных областях (гидравлика, аэродинамика, строительная механика, электродинамика и т.д.) разработаны свои системы критериев подобия и накоплен специфический опыт их использования. Физическое моделирование часто используют для обоснования достоинств новых технических решений.

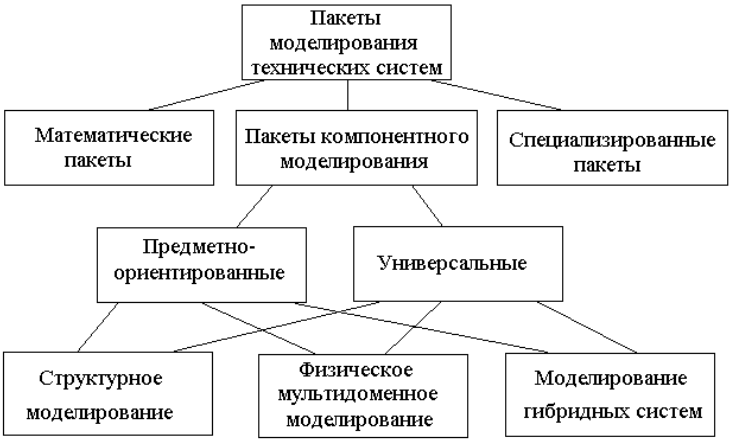


Рисунок 5.1 – Системы технического моделирования

Пакеты компонентного моделирования в основном ориентированы на численные эксперименты и являются в настоящее время доминирующими в процессах проектирования технических объектов. Они позволяют пользователю не заботиться о программной реализации модели, как о последовательности исполняемых операторов, и тем самым создают на компьютере некоторую удобную среду, в которой можно создавать виртуальные системы и проводить эксперименты с ними.

Пакеты компонентного моделирования, по способам их применения или технологии моделирования можно разделить на две группы. Так называемые универсальные пакеты ориентированы на определенный класс математических моделей и применимы для любой прикладной области, в которой эти модели справедливы. Основу универсального пакета составляют библиотеки компонентов общего назначения. В этих пакетах используются разнообразные коллекции численных методов, способные справиться с широким спектром задач. Как правило, универсальные пакеты обладают развитыми средствами визуализации, обеспечивающими показ изучаемого явления с разных сторон, а не одним, принятым в конкретной области способом.

Предметно-ориентированные пакеты предназначены для решения промышленных и научно-исследовательских задач в конкретной предметной области. Библиотеки моделей компонентов таких пакетов содержат хорошо изученные и отлаженные модели из довольно узкой предметной области, которые лишь накапливаются, модифицируются и приспосабливаются для решения конкретных задач. В результате накопленная база моделей со временем приобретает большую ценность. Спектр методов решения задач проектирования также ограничен известными и хорошо отработанными инструментами, возможно, ориентированными на узкий класс задач, в эффективности и надежности которых у пользователей нет сомнений. Как правило, предметно-ориентированные пакеты требуют серьезных усилий для их освоения, а также знаний в конкретной предметной области. Стоимость этих пакетов достаточно высока, что обычно заставляет пользователя применять какой-либо один пакет в течение длительного времени, всячески расширяя его возможности.

Следует заметить, что между универсальными и предметно-ориентированными пакетами нет четкой границы. Часто разница лишь количественная. Добавление к универсальному пакету соответствующего набора специализированных модулей, прежде всего библиотек моделей компонентов, превращает этот пакет в предметно-ориентированную среду моделирования. Примером подобного подхода может служить появление предметных расширений пакета Simulink – SimPower, SimMechanic и т.п. Учитывая открытость системы, каждый пользователь может добавить к готовым моделям то, что ему нужно, создав собственную предметно-ориентированную среду.

По принципам представления исходной модели среди пакетов компонентного моделирования можно выделить две основные группы:

1. Пакеты структурного (или блочного) моделирования.

2. Пакеты физического мультидоменного моделирования.

Элементарные блоки пакетов структурного моделирования обладают направленным действием, последующий блок не влияет на предыдущий. К достоинствам этого подхода следует отнести, прежде всего, простоту создания не очень сложных моделей даже не слишком подготовленным пользователем. Другим достоинством является эффективность реализации элементарных блоков и простота построения эквивалентной системы. В то же время при создании сложных моделей приходится строить довольно громоздкие многоуровневые блок-схемы, не отражающие естественной структуры моделируемой системы. Наиболее известными представителями пакетов визуального «структурного моделирования» являются: MATLAB/Simulink, EASY5,VisSim.

Пакеты физического мультидоменного моделирования позволяют использовать как ориентированные, так и неориентированные компоненты и связи. Подход очень удобен и естестве-

нен для описания типовых блоков физических систем. К пакетам «физического моделирования» можно отнести: Multisim, DYNAST, 20-SIM; Dymola.

Некоторые авторы выделяют в качестве третьей группы пакеты, предназначенные для моделирования гибридных систем. Эти пакеты позволяют очень наглядно и естественно описывать мехатронные системы со сложной логикой переключений. К этому направлению относится пакет Shift, а также отечественный пакет Model Vision Studium.

Рассмотрим очень коротко возможности и особенности некоторых универсальных и достаточно распространенных пакетов визуального моделирования, которые могут быть использованы для моделирования мехатронных систем.

К числу универсальных, не ориентированных на конкретные прикладные области пакетов для моделирования технических систем можно отнести пакет MATLAB/Simulink, а также построенные по его образу и подобию пакеты VisSim, МВТУ.

Данные пакеты предназначены для моделирования и исследования динамических систем в широком понимании этого термина, включая и дискретные, и непрерывные, и гибридные модели. Их отличает относительная простота и интуитивная ясность входных языков в сочетании с разумными требованиями к мощности компьютеров.

5.1. **Конструирование объектов с использованием пакета** LabView.

LabView (англ. Laboratory Virtual Instrumentation Engineering Workbench) – это среда разработки и платформа для выполнения программ, созданных на графическом языке программирования «G» фирмы National Instruments (США). *LabVIEW* используется в системах сбора и обработки данных, а также для управления техническими объектами и технологическими процессами. Идеологически *LabVIEW* очень близка к SCADA-системам, но в отличие от них в большей степени ориентирована на решение задач не столько в области АСУ ТП, сколько в области АСНИ.

Графический язык программирования «G», используемый в *LabVIEW*, основан на архитектуре потоков данных. Последовательность выполнения операторов в таких языках определяется не порядком их следования (как в императивных языках программирования), а наличием данных на входах этих операторов. Операторы, не связанные по данным, выполняются параллельно в произвольном порядке.

Программа *LabVIEW* называется и является виртуальным прибором (англ. Virtual Instrument) и состоит из двух частей:

- блочной диаграммы, описывающей логику работы виртуального прибора;
- лицевой панели, описывающей внешний интерфейс виртуального прибора.

Виртуальные приборы могут использоваться в качестве составных частей для построения других виртуальных приборов.

Лицевая панель виртуального прибора содержит средства ввода-вывода: кнопки, переключатели, светодиоды, верньеры, шкалы, информационные табло и т.п. Они используются человеком для управления виртуальным прибором, а также другими виртуальными приборами для обмена данными.

В этом примере демонстрируется работа свойства Blinking управляющих элементов передней панели (рис. 5.2). Создается кнопка булевого индикатора, где к ее свойствам прикрепляются рычажки двух состояний «true – false». В данном примере задействовано три свойства объекта: видимость кнопки, мерцание кнопки и смена цветов. Также добавлена кнопка булевого индикатора, которая в состоянии «true» останавливает программу.

Рисунок 5.2 – Blinking

В данной программе демонстрируется, как программно осуществить удаление точек с графиков кривых (рис. 5.3). Создается блок-схема буферизированного сбора данных с помощью графа XY Graph. Находится этот граф в палитре Controls>>Graph Indicators. Данный граф отображает две кривые со случайно построенными горизонтальным и вертикальным графиками. Точки добавляются в массив с помощью функции, которая получает их произвольным методом. С помощью двух кнопочных манипуляторов мы можем выбрать любую точку из массива и удалить ее. Также внизу отображается поле состояния текущей точки.

Данная программа демонстрирует принципы работы с данными типа Waveform (рис. 5.4). При рассмотренни типа графика Waveform Char видно, что этот график имитирует работу самописца. Наш Waveform Char отображает две кривые (осциллограммы). Waveform Chart выводит скалярные данные (т.е. просто числа). Поэтому, чтобы вывести что-то на график – мы подключаем к его терминалу два объекта. Чтобы данные поступали непрерывно, создаем внутри объектов бесконечные циклы. Для одновременного показа двух осциллограмм, объекты необходимо поместить в кластер. Находится этот график в палитре Controls>>Graph Indicators. Добавлены две булевские кнопки с указанием свойств видимости осциллограмм (можем отображать или скрывать кривые).

LabVIEW – это кроссплатформенная графическая среда разработки приложений. *LabVIEW* – в принципе универсальный язык программирования. И хотя этот продукт порой тесно связан с аппаратным обеспечением National Instruments, он тем не менее не связан с конкретной машиной. Существуют версии для Windows, Linux, MacOS. Исходные тексты переносимы, а программы будут выглядеть одинаково во всех системах. Код, сгенерированный *LabVIEW* также может быть исполнен на Windows Mobile или PalmOS. Этот язык может с успехом использоваться для создания больших систем, для обработки текстов, изображений и работы с базами данных *LabVIEW* – весьма высокоуровневый язык. Однако ничто не мешает включать «низкоуровневые» модули в *LabVIEW*-программы. Даже если вы хотите использовать ассемблерные вставки – это тоже возможно, надо

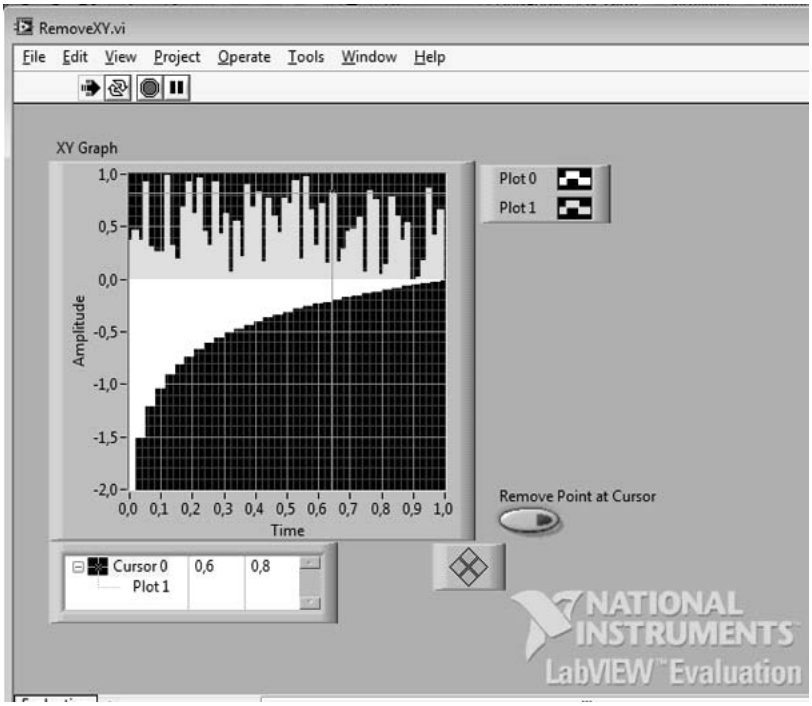


Рисунок 5.3 – XY Graph

лишь сгенерировать DLL и вставить вызовы в код. С другой стороны, высокоуровневый язык позволяет запросто производить весьма нетривиальные операции с данными, на которые в обычном языке могли уйти многие строки (если не десятки строк) кода. Впрочем, ради справедливости надо отметить, что некоторые операции низкоуровневых языков (например, работу с указателями), не так просто реализовать в *LabVIEW* ввиду его «высокоуровневости». Разумеется, язык *LabVIEW* включает основные конструкции управления, имеющие аналоги и в «традиционных» языках:

- переменные (локальные или глобальные);
- ветвление (case structure);
- For – циклы с проверкой завершения и без;
- While – циклы;
- группировка операций.

В LabVIEW разрабатываемые программные модули называются «Virtual Instruments» (виртуальные инструменты) или по-простому VI. Они сохраняются в файлах с расширением *.vi. VIs – это «кирпичики», из которых состоит LabVIEW – программа. Любая LabVIEW программа содержит как минимум один VI. В терминах языка Си можно достаточно смело провести аналогию с функцией с той лишь разницей, что в LabVIEW одна функция содержится в одном файле (можно также создавать библиотеки инструментов). Само собой разумеется, один VI может быть вызван из другого VI. В принципе каждый VI состоит из двух частей – блок-диаграмма (Block Diagram) и передняя панель (Front Panel). Блок-диаграмма – это программный код (точнее визуальное графическое представление кода), а передняя панель – это интерфейс. Вот как выглядит классический пример Hello, World!: (рис. 5.5).

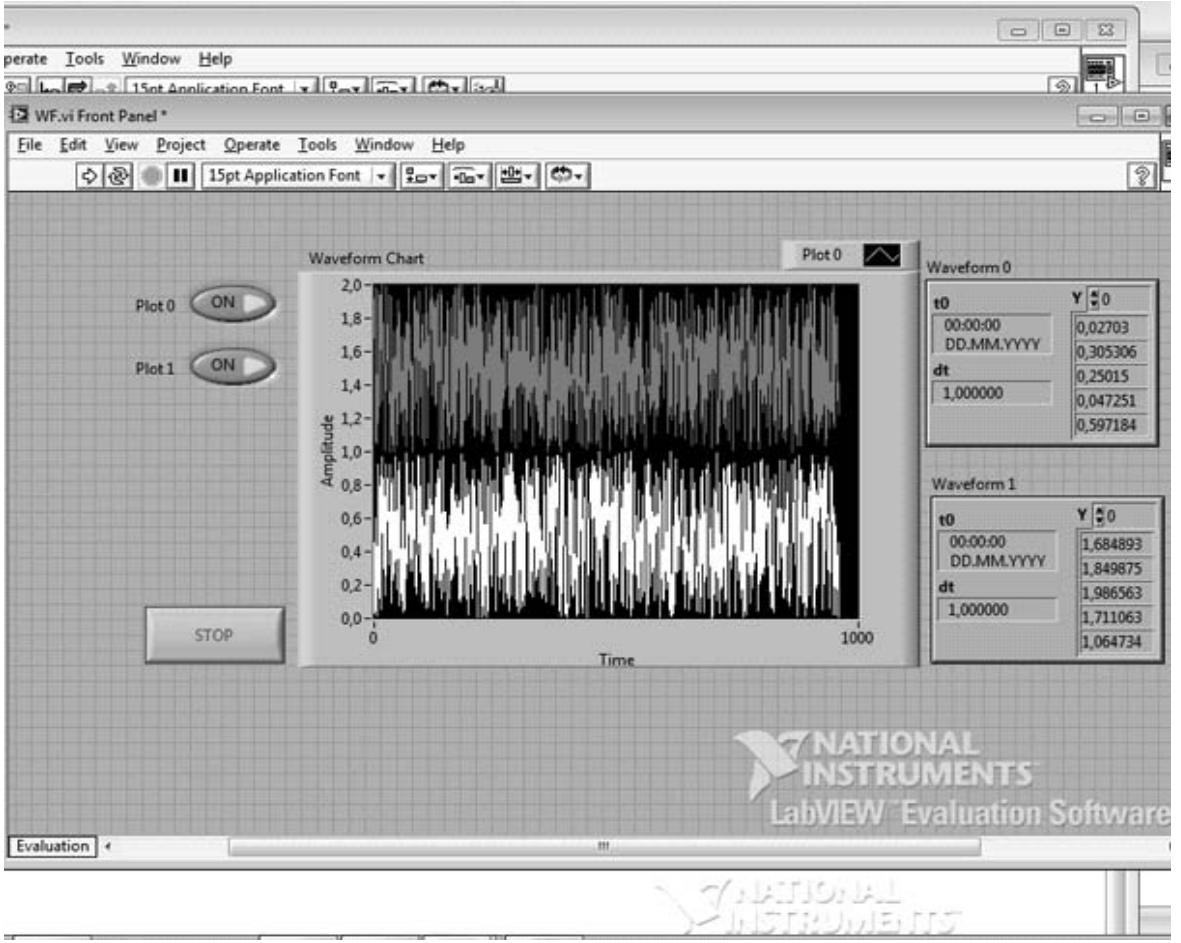


Рисунок 5.4 – Waveform Char

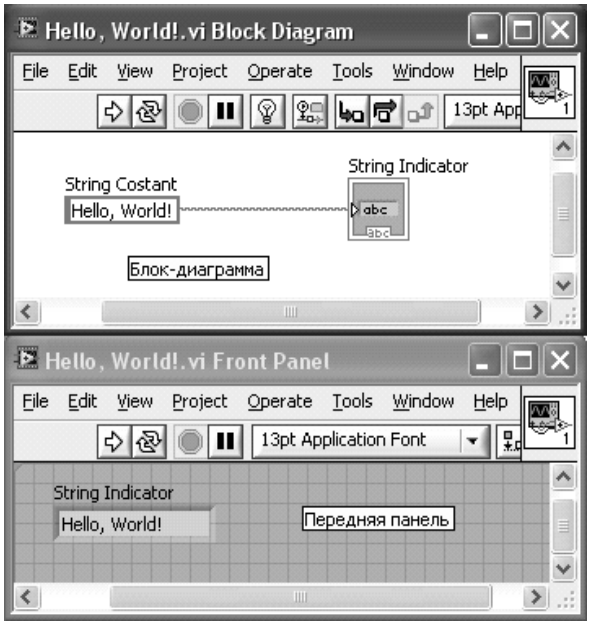
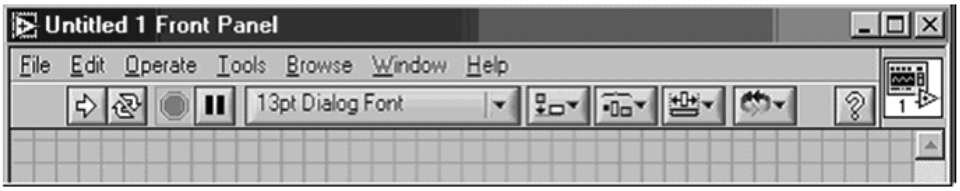
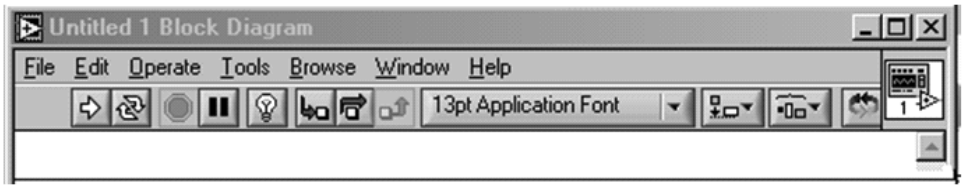


Рисунок 5.5 – Блок диаграмма примера Hello, World

Для создания собственных программ в среде LabVIEW (рис. 5.7) используются следующие инструменты: лицевая панель, блок-диаграмма, палитры элементов управления и отображения данных и палитры функций. При запуске *LabVIEW* из меню стартового диалогового окна командами New Blank VI открываются два окна – лицевая панель и блок-диаграмма (рис. 5.6) . В правом верхнем углу каждого окна находится пиктограмма для архивирования созданной программы в качестве нового компьютерного прибора. Здесь же размещена традиционная для приложений Windows полоса главного меню с одинаковыми для обоих окон пунктами: File, Edit, Operate, Tools, Browse, Windows, Help.



а



б

Рисунок 5.6 – Главное меню

В среде *LabVIEW* (рис. 5.7) используются различные типы данных:

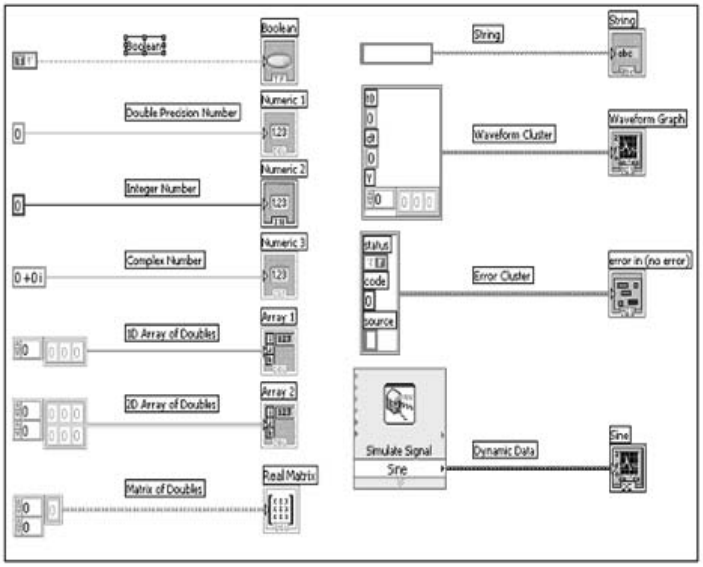


Рисунок 5.7 – Среда LabVIEW

*5.1.1. **Конструирование объектов с использованием пакета** LabVIEW*

Рассмотрим возможности среды для быстрого создания профессионального интерфейса и обработки данных на примере разработки программы моделирования политропного процесса сжатия воздуха.

Пускай результаты вычислений необходимо отобразить в виде индикаторов традиционных приборов, служащих для измерения *V, P, T,* графиков их изменения по времени и *P-V* диаграмм исследованного процесса. Как известно, существует ряд процессов, в течение которых сохраняется постоянное отношение выполненной работы и количества тепла, участвующего в теплообмене с внешней средой. Такие процессы называются политропными.

Лицевая панель пользователя, с которой осуществляется управление процессом моделирования. В ее верхней части находится четыре цифровых элемента управления для введения исходных данных задачи: *V0, P0, T0, n*.

Для отображения текущих значений *V, P, T* на лицевой панели помещены четыре виртуальных прибора – мерная емкость, стрелочный манометр, термометр и цифровой секундомер с верхними пределами показаний их шкал, соответствующими диапазонам измерений объема – 1000 мл, давления – 2000 кПа, температуры 1000 К и 1000 секунд виртуального времени. Показания линейных шкал этих приборов продублированы цифровыми индикаторами, позволяющими производить более точный отсчет контролируемых параметров.

Для наблюдения за ходом моделируемого процесса на лицевой панели (рис. 5.8) находятся виртуальный трехлучевой запоминающий осциллограф и *X-Y*-самописец для построения *P-V* диаграммы процесса.

Графический код программы моделирования (рис. 5.9):

В центре блок-схемы, которая для удобства разработки программы присутствует на экране компьютера одновременно с лицевой панелью, находится графическая пиктограмма цикла по условию в виде внешнего прямоугольника, заключающего все операторы, выполняющиеся внутри этого

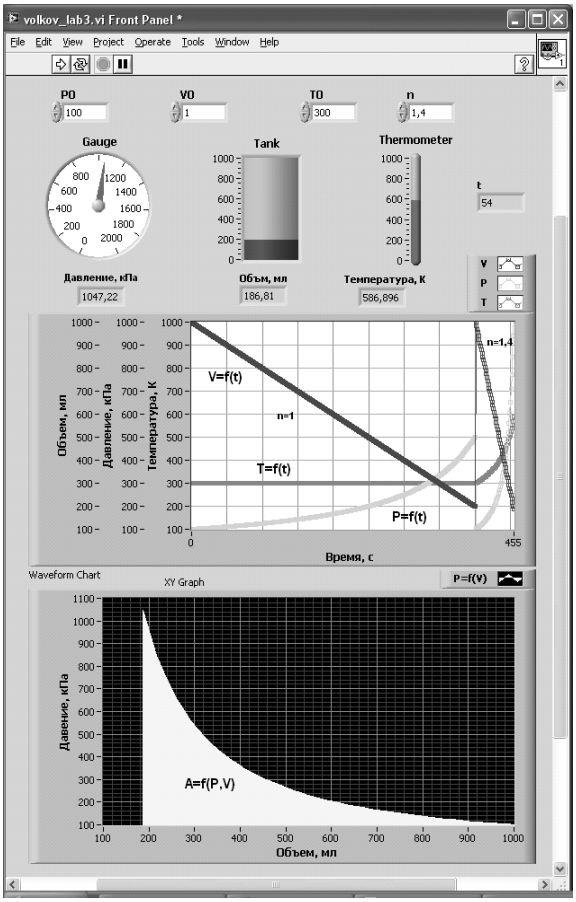


Рисунок 5.8 – Лицевая панель

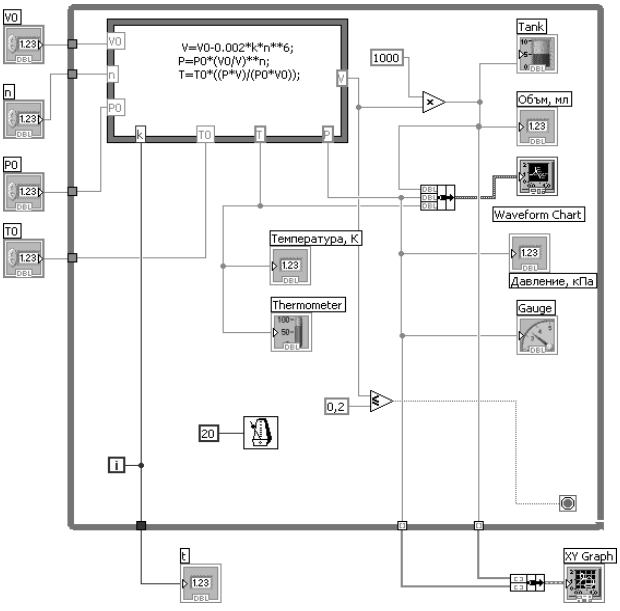


Рисунок 5.9 – Графический код программы

цикла. Следующей центральной структурой программы является узел формул, в который в обычной записи внесены основные соотношения математической модели, определяющие изменения параметров состояния газа в зависимости от скорости его сжатия и виртуального времени *k*. Для обеспечения работы операторов формульного узла необходимо ввести через терминалы входа значения всех переменных, присутствующих в левой части записанных уравнений. Вычисленные значения параметров (правые части формул) выводятся через терминалы выхода, выделенные более жирным контуром.

Программа автоматически прекращает работу, когда по условию задачи степень сжатия воздуха в цилиндре $\lambda = V_0/V_K$ становится равной или больше пяти.

В основе *LabVIEW* лежит парадигма потоков данных. В выше приведённом примере константа и терминал индикатора соединены между собой линией. Эта линия называется Wire. Можно назвать её «проводом». По проводам передаются данные от одних элементов другим. Вся эта концепция называется Data Flow. Суть блок диаграммы – это узлы (ноды), выходы одних узлов присоединены к входам других узлов. Узел начнёт выполнение только тогда, когда прибудут все необходимые для работы данные. На диаграмме вверху две ноды. Одна из них – константа. Этот узел самодостаточен – он начинает выполнение немедленно. Второй узел – индикатор. Он отобразит данные, которые передаёт константа.

Алгоритм решения: сложение и умножение двух чисел.

int a, b, sum, mul; //... sum = a + b; mul = a * b;

Пример решения этой же задачи в MatLab (рис. 5.10).

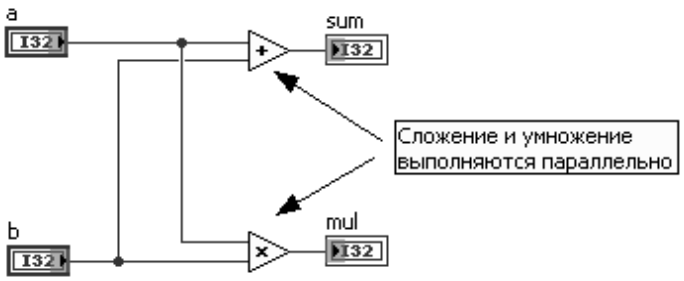


Рисунок 5.10 – Пример решения задачи

А вот как выглядят while/for циклы и if/then/else структура (рис. 5.11):

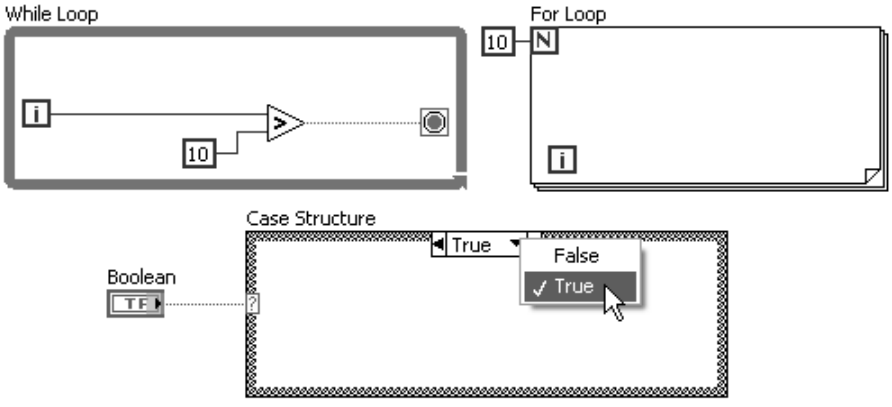


Рисунок 5.11 – Цикл и структура

5.2. **Конструирование объектов с использованием пакета** Vissim

В мире информационных технологий компьютерное моделирование переживает второе рождение. Интерес к компьютерному моделированию оживился в связи с существенным технологическим развитием систем моделирования, которые на сегодняшний день являются мощным аналитическим средством, воплотившим в себя весь арсенал новейших информационных технологий, включая развитые графические оболочки для целей конструирования моделей и интерпретации выходных результатов моделирования, объектно-ориентированный подход, и т.д.

На сегодняшний день на рынке существует множество пакетов компьютерных программ, предназначенных для компьютерного моделирования систем. В связи с этим перед человеком, предполагающим использовать компьютерное моделирование в качестве инструмента исследования, неминуемо встаёт проблема выбора. AnyLogic, Arena, AutoMod, Dymola, Extend, eM-Plant, EASY5, Gpss/H-Prof, iThink, Matlab, Modelica, ModelVision, MBTY, ProModel, PowerSim, Quest, Stella, Simfactory, Simplle++, Taylor, Vensim, VisSim, Witness – вот далеко не полный список продуктов, каждый из которых претендует на лидерство. Часть продуктов позиционируют себя как пакеты общего назначения, которые могут использоваться как аналитиками-экономистами, так и инженерами-проектировщиками, другая часть продуктов специализируется на какой-либо узкой области (например, моделирование производственного процесса). Некоторые из этих продуктов появились весьма недавно, другие считаются классическими, и все они используют различные подходы к решению одних и тех же задач. К счастью, в основе этих подходов, зачастую, лежат результаты классических фундаментальных работ в области моделирования (понятие гибридного автомата, карты состояний Харэла, концепции объектно-ориентированного программирования и т. д.).

Если «рассмотреть» пакет VisSim (от Visual Solutions Inc.), то под словом «рассмотрение» следует понимать не просто ознакомление с возможностями пакета (входным языком, инструментами анализа), а скорее ответы на такие вопросы как:

- какой класс задач покрывает функциональность VisSim (а какой отдаётся на откуп другим пакетам);
- *VisSim* – классический пакет, на какое место среди современных пакетов он может претендовать, каковы его сильные и слабые стороны?

Фирма Visual Solutions является производителем такого программного продукта как VisSim с 1989 года. VisSim – это диалоговая визуальная оболочка для разработки непрерывных, дискретных, мультичастотных и гибридных моделей систем и моделирования динамики этих систем. Набор команд, предоставляемый VisSim, позволяет автоматизировать решения многих задач.

В VisSim модель системы сроится в виде структурной схемы в «привычном виде», т.е. в виде понятном для инженеров-проектировщиков, владеющими понятиями теории автоматического управления, пакет изначально позиционировался как система моделирования технических и физических систем. Одновременно с VisSim, MathWorks выпустило Mathlab Simulink, оба продукта – коммерческие, и оба претендуют на звание пионера в использовании языка иерархических структурных схем (иначе, «сигнальных графов»). Mathlab Simulink получил несколько большее распространение, тем не менее, некоторые особенности пакета VisSim позволили ему получить довольно большую долю рынка инженерных пакетов.

Основными инструментами являются функциональные блоки и связи между ними. Каждый блок выполняет определенную функцию. Функция может быть такой же простой как «sin» или сложной, как передаточная функция 10-го порядка. VisSim содержит более 100 линейных и нелинейных блоков, позволяющих моделировать сколь угодно сложные системы. Но если вдруг по каким-то причинам предоставленный набор средств окажется недостаточным, то VisSim предоставляет простой механизм расширения за счет пользовательских блоков.

VisSim имеет частотные, корневые, вариационные, нейронные инструменты оценки качества, устойчивости, синтеза, коррекции, оптимизации, линеаризации, отладки объектов в контуре модели и программирования цифровых сигнальных процессоров.

VisSim имеет решатель интерпретирующего типа, функционирующий в динамическом режиме с возможностью online-взаимодействия с оборудованием реального времени. В состав пакета решателя VisSim-а входят: явные решатели – для решения дифференциальных уравнений, неявные – для решения алгебраических уравнений, а также оптимизаторы – для итерационного подбора параметров. Интерпретатор VisSim-а позволяет автоматически создавать С-код промышленного качества (в том числе с фиксированной точкой для цифровых сигнальных процессоров). Возможности управления потоком исполнения модели заключены в свободном выборе величин локальных шагов симуляции (для НЧ-фрагментов модели), и в программировании серии повторных симуляций (либо для оптимизации, либо для изучения поведения модели в условиях случайных возмущений). Для поддающихся линеаризации фрагментов модели *VisSim* выполняет следующие виды символьного анализа: определение коэффициентов передаточной функции и ABCD-матриц пространства состояний, определение нулей и полюсов передаточных функций, билинейное преобразование (переход от линейных систем к дискретным и обратно). Опираясь на результаты линеаризации модели, VisSim выполняет корневой анализ (годограф корней) и частотный (ЛАЧХ&ЛФЧХ, годограф Найквиста). Так же *VisSim* имеет мастера для генерации коэффициентов классических линейных фильтров (Бесселя, Баттерворта, Чебышева, инверсного Чебышева), и дискретных (КИХ, БИХ-фильтров, преобразователя Гильберта, дифференциатора). Базовая библиотека блоков VisSim-а (в списке менее 100 позиций) не требует дальнейшего расширения, хотя пользователю предоставлена возможность определить собственную библиотеку моделей.

Инструментальные средства (пакеты) для моделирования (рис. 5.12) можно разделить на две группы. К первой относятся специализированные программные средства (пакеты), ориентированные на специфические понятия конкретной прикладной области (химической технологии, теплотехники, электротехники и т.д.). Ко второй относятся так называемые «универсальные» пакеты, ориентированные на определенный класс математических моделей и применимые для любой прикладной области, в которой эти модели пригодны. Ясно, что специализированные пакеты могут быть использованы только для автономного исследования отдельных подсистем сложной динамической системы.

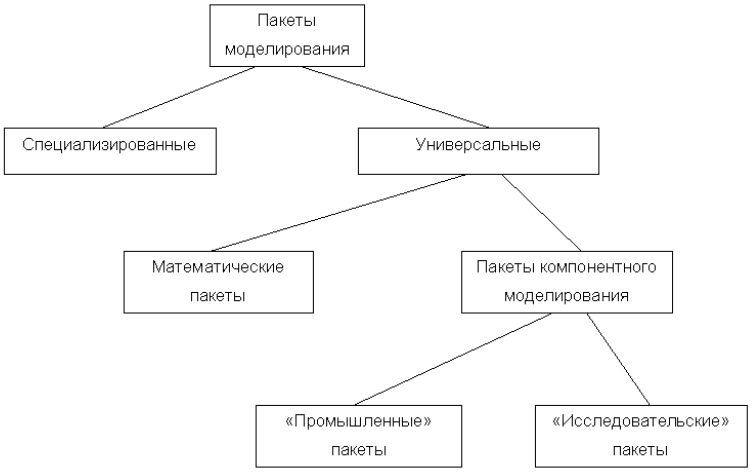


Рисунок 5.12 – Инструментальные средства (пакеты) для моделирования

Полученные специализированные модели чрезвычайно сложно использовать в дальнейшем для построения комплексной модели сложной динамической системы, а также для отработки программ на этапе проектирования. Поэтому даже при моделировании отдельных подсистем предпочтительно использовать «универсальные» пакеты.

«Универсальные» пакеты далее обычно разделяют на «математические» пакеты и пакеты компонентного моделирования. В «математических» пакетах (Mathematica, MathCAD, MatLab, Maple) предполагается, что математическая модель всей моделируемой системы уже каким-либо образом построена и ее требуется исследовать. Такой подход характерен в основном для научных исследований. Как правило, математические пакеты сочетают численные эксперименты с символьными преобразованиями. Компонентное моделирование предполагает, что описание моделируемой системы строится из компонентов (в том числе и готовых библиотечных), а совокупная математическая модель формируется пакетом автоматически. Пакеты компонентного моделирования в основном ориентированы на численные эксперименты. Компонентное моделирование преобладает в процессе проектирования технических объектов. Очевидно, что для сложных динамических систем построение и сопровождение ее полной математической модели вручную практически невозможно. Следовательно, для аналитического моделирования должны использоваться пакеты компонентного моделирования, которые по способам их применения или технологии моделирования также можно разделить на две группы.

К первой отнесем пакеты, предназначенные для решения сложных промышленных и научно-исследовательских задач большими производственными или научными коллективами. В таких проектах ведущую роль играет организация работ: хорошо налаженное взаимодействие между отдельными группами, быстрый доступ к многочисленным экспериментальным данным и библиотекам программ, тщательное документирование и тестирование, многовариантные расчеты. При этом обычно используются хорошо изученные готовые математические модели, которые лишь модифицируются и приспосабливаются для решения конкретных задач. Пользователи пакета подразделяются на две категории: разработчики библиотек готовых моделей и обычные пользователи, работа которых сводится к составлению схем из типовых блоков и параметрическая настройка блоков. Пакеты первой группы условно назовем «промышленными».

Совсем другая технология характерна для предварительных исследований, выполняемых отдельными учеными или проектировщиками. Библиотеки готовых моделей используются весьма ограниченно. Исходным материалом служат плохо формализованные «сырые» модели, то есть модели, чьи свойства еще не вполне осознаны. Это означает, что необходимо уметь организовывать и поддерживать непрерывную обратную связь между исследователем и исследуемой моделью. Несмотря на большие достижения в области автоматического синтеза систем с заданными показателями, на практике разработка новой технической системы – это прежде всего просмотр большого числа пробных вариантов. Назовем пакеты второй группы «исследовательскими», подчеркивая этим, что они уступают по количеству уникальных возможностей промышленным, зато более просты для освоения и доступны отдельному исследователю при решении относительно несложных задач из практически любой прикладной области. Под «несложными» будем понимать не простые задачи, а задачи посильные одному разработчику, не являющемуся специалистом в области программирования и вычислений.

Несмотря на то, что *VisSim* относится к универсальным пакетам моделирования, его функциональность значительно расширена множеством специализированных библиотек, о которых поговорим чуть позже. *VisSim* – это пакет именно компонентного моделирования, в котором модель задаётся при помощи стандартных блоков, параметры которых и определяют коэффициенты в уравнениях математической модели. Можно поспорить: является *VisSim* промышленным или исследовательским пакетом. Общепринята в этом вопросе следующая позиция. *VisSim* –

«промышленный» пакет, так как обладает множеством библиотек, на основе которых созданы стандартные модели, помогающие решать специализированные задачи для каждой прикладной области. Разработка этих библиотек практически невозможна на уровне пользователя, так как пользователь лишь имеет возможность создавать новые блоки на уровне языка моделирования, объединяя другие блоки в более сложные подсистемы, в то время как большинство стандартных библиотек написана на языке более низкого уровня (например, языке программирования C++), и их написание требует от пользователя высокой квалификации программиста.

Входной язык, возможности его расширения пользователем.

В современных инструментах компонентного моделирования можно выделить два основных направления:

- «блочное моделирование»;
- «физическое моделирование».

Системы, базирующиеся на первом подходе, ориентированы на графический язык иерархических блок-схем. Системы, использующие второй подход, характеризуются отказом от ориентированных связей между компонентами. На смену этим связям приходят агебро-дифференциальные уравнения, задающиеся в достаточно свободной форме.

VisSim – один из наиболее ярких представителей систем, реализующих концепцию блочного моделирования. Типовые блоки «вход-выход-состояние» – основной инструмент задания моделей, каждый такой типовой блок реализует математическую модель или визуализацию того или иного явления, процесса или устройства. Подобный язык задания графических функциональных схем восходит к схемам, применявшимся ещё на аналоговых машинах.

Использовать предопределенные блоки — это выгодно, если структура модели проста и все необходимые блоки уже присутствуют в базовом наборе. Набирать же произвольную сложную систему уравнений с помощью интеграторов, сумматоров и усилителей подобно тому, как это делалось на аналоговой машине, неудобно. Графический образ модели становится ненаглядным и, главное, не отражает естественной структуры моделируемого объекта.

Один из наиболее важных недостатков «блочного» подхода к моделированию – невозможность полноценного задания новых блоков. Большинство из используемых блоков написаны с использованием языка программирования C++. Интерфейс пользователя VisSim позволяет задавать новые блоки (подсистемы), основываясь на уже существующих блоках. При существующем подходе мы не сможем создать, к примеру, собственный интегратор, так как наш интегратор не может быть представлен с использование уже существующих блоков. Создатели *VisSim* предусмотрели такую возможность. Мы можем создавать свои собственные библиотеки практически на любом процедурном языке программирования, и подключать созданную библиотеку в виде DLL модуля. При использовании C++, существует упрощённая процедура задания новых блоков с помощью надстройки *VisSim/C-Code.*

Стандартные блоки сгруппированы по функциональному признаку, всего 18 групп:

- Animation – блоки animate и lineDraw, позволяющие организовать простейшую анимацию;
- Annotation – блоки, позволяющие организовать комментарии и аннотацию к модели;
- Arithmetic – блоки арифметических операций;
- Audio – блоки, позволяющие работать со звуком, сохраненыв идее WAV файла;
- Boolean – блоки, имплементирующие логические операции;
- DDE – блоки, позволяющие обмениваться информацией с другими Windows приложениями;
- Integration – простейшие интеграторы;
- Linear System — блоки облегчающие работу с линейными системами;
- Matrix Operations – блоки, реализующие матричные операции;
- Nonlinear – блоки, упрощающие работу с нелинейными системами;

- Optimization – блоки, позволяющие организовать оптимизационный эксперимент над созданной моделью;
- Random Generator – три блока, задающих три различных распределения случайной величины;
- Real Time – блоки, служащие для интеграции с внешними приборами;
- Signal Consumer – блоки для вывода и визуализации сигналов;
- Signal Producer – источники сигналов;
- State Transition – блок позволяющий моделировать набор дискретных состояний;
- Time Delay – блоки организации задержки;
- Transcendental – встроенные тригонометрические функции.

Для создания собственных блоков предусмотрены блоки:

- emded – блок, позволяющий встраивать одну модель в другую;
- userFunction – блок, позволяющий работать с библиотеками блоков, заданных в виде DLL;
- expression – блок позволяющий вставлять код на языке C;
- OLEobject – блок позволяющий работать с OLE объектами.

Для начала откроем заранее скачанную программу VISSIM и создадим на рабочей области 2 синусоидальных источника сигнала (рис. 5.13).

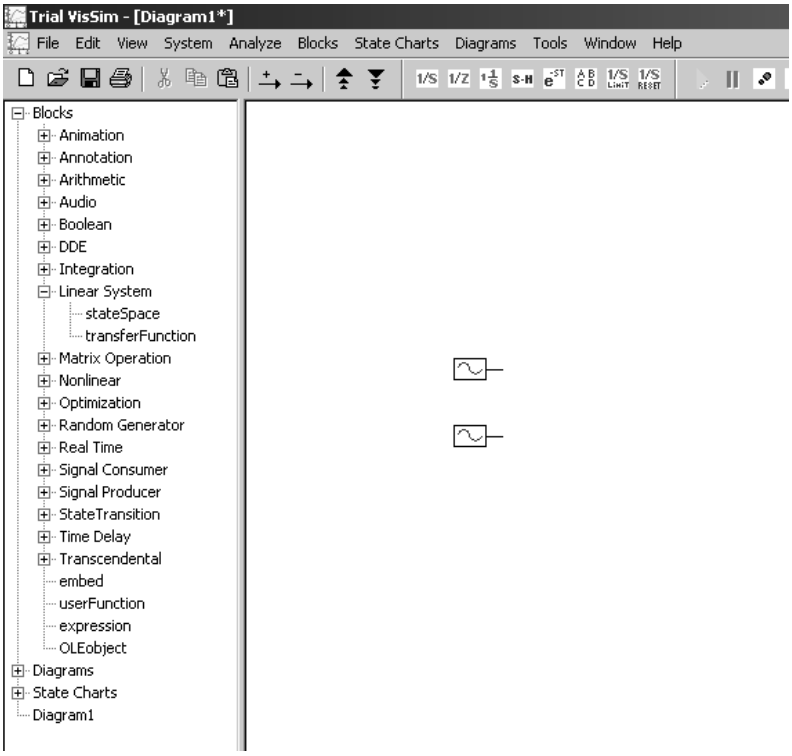


Рисунок 5.13 – Создание двух источников сигнала

Затем в арифметических блоках найдем блок отрицания и вытащим его на рабочую область. В блоках вывода сигнала найдем plot 2D и аналогичным образом перетащим его на рабочую область VISSIMA. После соединим все блоки так, как показано на картинке ниже (рис. 5.14).

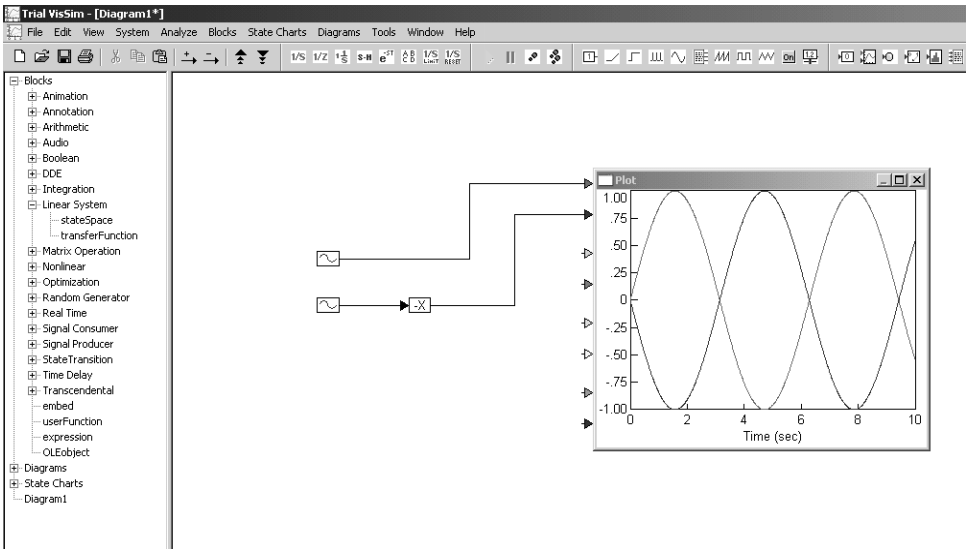


Рисунок 5.14 – Подключение их к блоку вывода plot 2D

В результате получим 2 синусоидальных сигнала. Один из них перевернут, из-за влияния блока отрицания. Теперь добавим булевый блок (рис. 5.15).

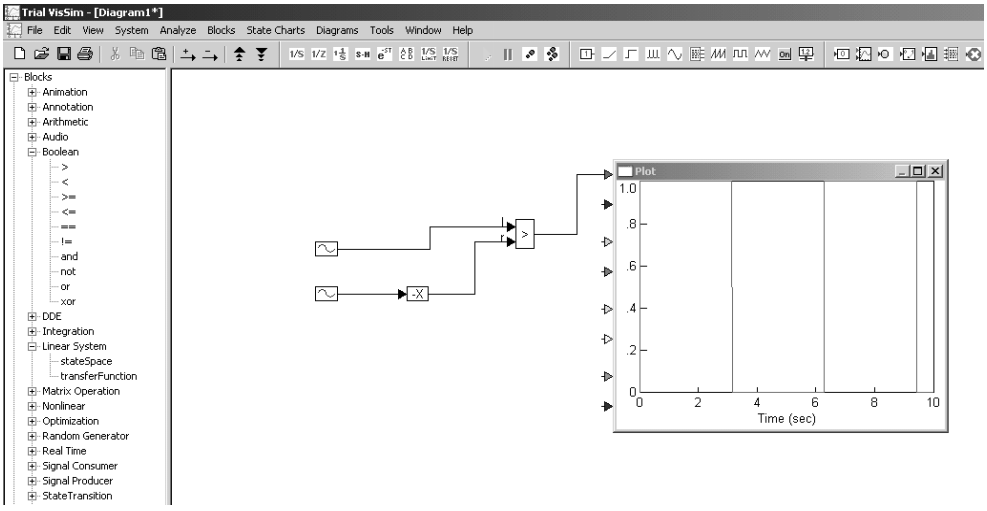


Рисунок 5.15 – Добавление булевого блока

В результате получился график сравнения двух синусоидальных функций относительно времени.

5.3. **Конструирование объектов с использованием пакета** DesignLab 8.0

На платформе персональных компьютеров в настоящее время имеется достаточно много систем, обеспечивающих сквозное проектирование радиоэлектронной аппаратуры. Наибольшее распространение среди разработчиков радиоэлектронной аппаратуры получила система *DesignLab*

корпорации MicroSim. Основу системы *DesignLab* составляет программа PSpice, которая является наиболее известной модификацией программы схемотехнического моделирования SPICE (Simulation Program with Integrated Circuit Emphasis), разработанной в начале 70-х годов в Калифорнийском университете г. Беркли.

В состав системы *DesignLab* входят следующие программы:

- Schematics – графический редактор принципиальных схем, который одновременно является управляющей оболочкой для запуска основных модулей системы, на всех стадиях работы с проектом;
- PSpiceA/D – моделирование смешанных аналого-цифровых устройств;
- PLSyn – синтез цифровых устройств на базе интегральных схем (ИС) с программируемой логикой PLD/CPLD;
- StmEd– редактор входных сигналов (аналоговых и цифровых);
- Probe – графическое отображение, обработка и документирование результатов моделирования;
- Parts – идентификация параметров математических моделей диодов, биполярных, полевых и мощных МОП-транзисторов, биполярных статически индуцированных транзисторов, операционных усилителей, компараторов напряжения, регуляторов и стабилизаторов напряжения и магнитных сердечников по паспортным данным;
- PC Boards и Autorouter – графический редактор многослойных печатных плат и программа автотрассировки SPECCTRA фирмы Cadence (рассчитаны на 6 сигнальных слоев);
- Micro Sim FPGA – интерфейс с программой XACT Step 6.0, предназначенной для проектирования электрически перепрограммируемых ПЛИС Фирмы Xilinx;
- PSpice –моделирование аналоговых устройств;
- PSpice Optimizer – параметрическая оптимизация аналого-цифровых устройств по заданному критерию при наличии нелинейных ограничений;
- Polaris – проверка целостности сигнала, т.е. проведение моделирований с учетом паразитных емкостей и индуктивностей, присущих реальным печатным платам;
- PLogic – моделирование цифровых устройств;
- Device Equations – исходный текст встроенных математических моделей полупроводниковых приборов на языке Си. В них можно изменять имена параметров, вводить псевдонимы, добавлять параметры и модифицировать уравнения моделей.

К пакету *Design Lab* прилагаются библиотеки примерно 40 тыс. графических обозначений символов, около 10 тыс. математических моделей компонентов (диодов, стабилитронов, тиристоров, биполярных и полевых транзисторов, оптопар, операционных усилителей, компараторов напряжения, стабилизаторов напряжения, кварцевых резонаторов, магнитных сердечников, цифровых и аналого-цифровых ИС) и 1000 корпусов компонентов производства фирм США, Западной Европы и Японии.

Работа с системой *DesignLab* (рис. 5.16) обычно начинается с создания принципиальной схемы с помощью редактора схем *Schematics* (рис. 5.17). Он позволяет создавать чертежи принципиальных схем и осуществлять запуск других программ (*PSpice, Probe* и др.). Моделируемая схема может состоять из следующих типовых компонентов: резисторы, конденсаторы, индуктивности, трансформаторы (в том числе с магнитными сердечниками), диоды (включая стабилитроны и варикапы), биполярные, арсенид-галлиевые, полевые, МОП – и биполярные статически индуцированные транзисторы, ключи, управляемые током и напряжением, линии передачи с потерями, аналого-цифровые и цифроаналоговые преобразователи, цифровые элементы (вентили, триггеры, устройства контроля, запоминающие устройства и программируемые логические матрицы). Кроме того, из этих компонентов можно создать макромодели или иерархические структуры более сложных компонентов устройств, например операционных усилителей, компараторов напряжения, регуляторов напряжения и др.

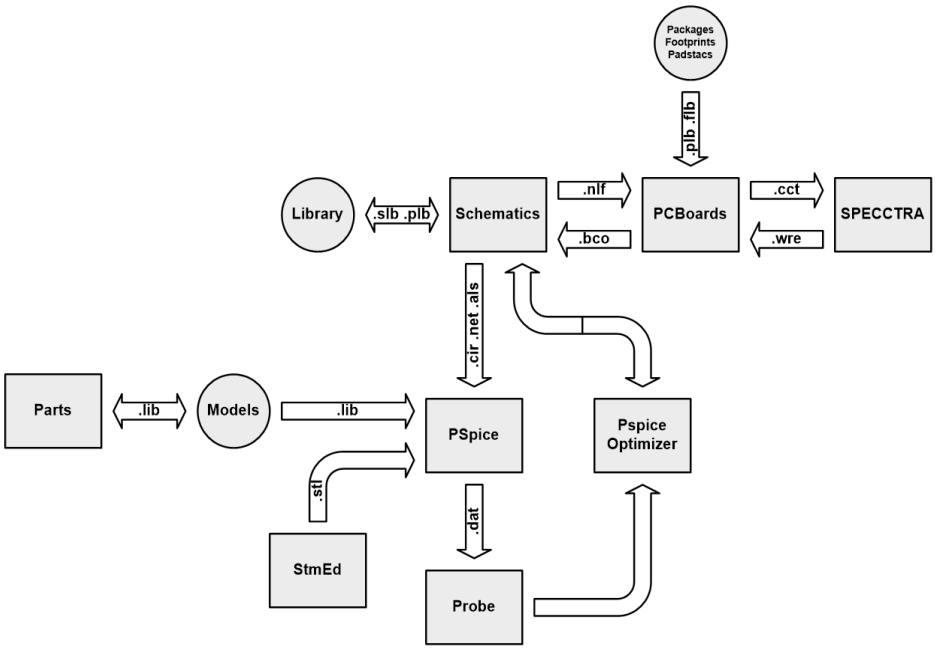


Рисунок 5.16 – Структурная схема системы Design Lab

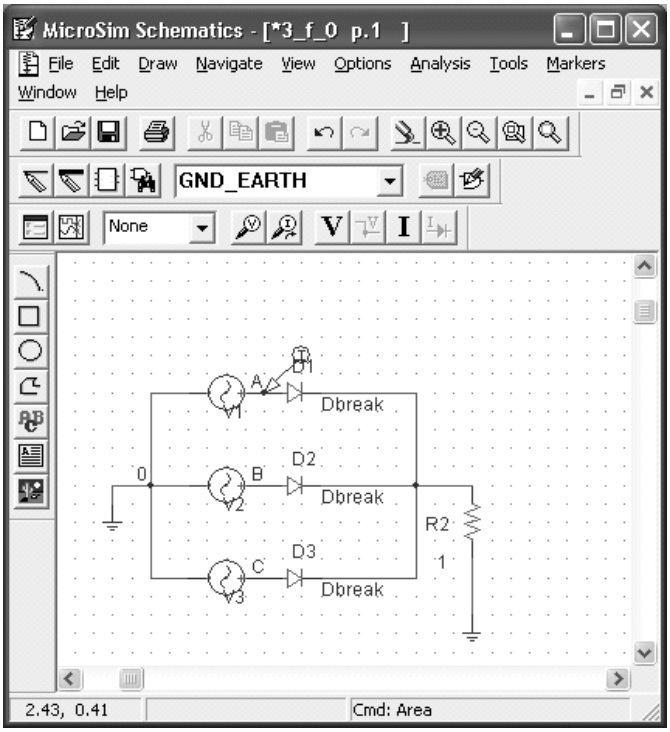


Рисунок 5.17 – Окно программы *Schematics*

Графический редактор вызывается щелчком левой кнопки мыши по пиктограмме *Schematics* в меню Пуск\Программы\ Design Lab Release 8.0\Schematics. В процессе его загрузки подключаются библиотеки графических символов компонентов.

Редактор *Schematics* имеет основное меню и панель инструментов. Встроенная помощь позволяет получать краткую информацию об основных правилах работы (раздел Help в меню команд). На схему можно наносить символы кириллицы, применяя шрифты *TrueType*.

В верхней части экрана располагается горизонтальное меню, состав пунктов которого зависит от выбранного режима редактирования (редактирование принципиальных электрических схем либо редактирование символов, т.е. условных графических обозначений компонентов).

Вначале нужно выбрать курсором команду File, после чего в ниспадающем меню выбрать строку New , если создается новая схема, или строку Open, если загружается существующая схема.

Символы компонентов наносятся на схему по команде Draw\Get New Part (а также щелчком по пиктограмме). Вращение символа удобно производить комбинацией клавиш [*Ctrl*] + [*R*], зеркальное отображение – [*Ctrl*] + [*F*]. Последние выбираемые символы удобно наносить из открывающегося списка графического меню. Для каждого компонента можно задать один или несколько параметров (выбрав его двойным щелчком левой кнопки мыши), перечень которых указывается заранее при создании его условного графического обозначения. Конкретные значения параметров задаются с помощью диалогового окна редактирования атрибутов. Соединительные провода наносят «карандашом» – щелчок по пиктограмме *Drawwire* .

Для подготовки схемы к моделированию необходимо в среде редактора Schematics задаться директивами моделирования по команде Analysis\Setup, где устанавливается временной интервал и шаг подсчета параметров моделирования.

Программа моделирования PSpice запускается командой Analysis\Simulate или нажатием на клавишу [*F11*]. В верхней части экрана помещена строка меню для загрузки файлов, изменения цвета и шрифта, вызова подсказки необходимо открыть файл с тем же именем, что у схемы.

Далее в окне программы PSpice (рис. 5.18) выводится название задания на моделирование, имя файла, название выполняемой команды и значения варьируемых параметров, температуры и др. В средней части экрана указывается название вида анализа и приводятся информационные сообщения о фазах выполнения задания на моделирование и сообщения об ошибках (последние выделяются красным цветом). В нижней части экрана в реальном масштабе времени выводятся текущие значения от одной до трех переменных, перечисленных в директиве *WATCH* [на схеме эта директива задается установкой символа в виде глаза], и информация о шаге изменения независимых переменных и диапазоне их значений.

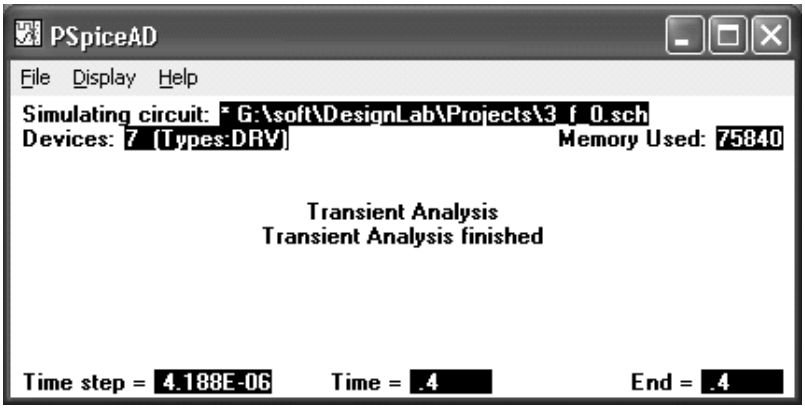


Рисунок 5.18 – Окно программы PSpice

Для вывода результатов моделирования используется программа *Probe*. Она выводит на экран графики результатов моделирования, производит их математическую обработку и выводит на экран в табличной форме их важнейшие характеристики, наносит на графики поясняющие надписи и позволяет получать жесткие копии результатов моделирования в графической форме.

Математические преобразования над графиками заключаются в выполнении арифметических операций, вычислении различных функций, взятии интегралов, расчете спектров, гистограмм, измерении, параметров формы графиков, построении зависимостей любой характеристики графика от любого варьируемого параметра схемы.

Программа *Probe* вызывается автономно или под управлением *Schematics* выбором команды Analysis\Probe или нажатием клавиши [*F12*] (предварительно рекомендуется выполненную программу моделирования *PSpice* текущей схемы закрыть). При вызове *Probe* из программы *Schematics* можно организовать экран с несколькими окнами для изображения схемы и графиков различных характеристик.

В окне программы *Probe (*рис.5.19*)* изображены графики переходных процессов и частотных характеристик. Переменные, графики которых должны быть выведены на экран, указываются отметкой их специальными маркерами на окне схемы. Возможность переключения окон со схемой и графиками существенно облегчает осмысление результатов моделирования.

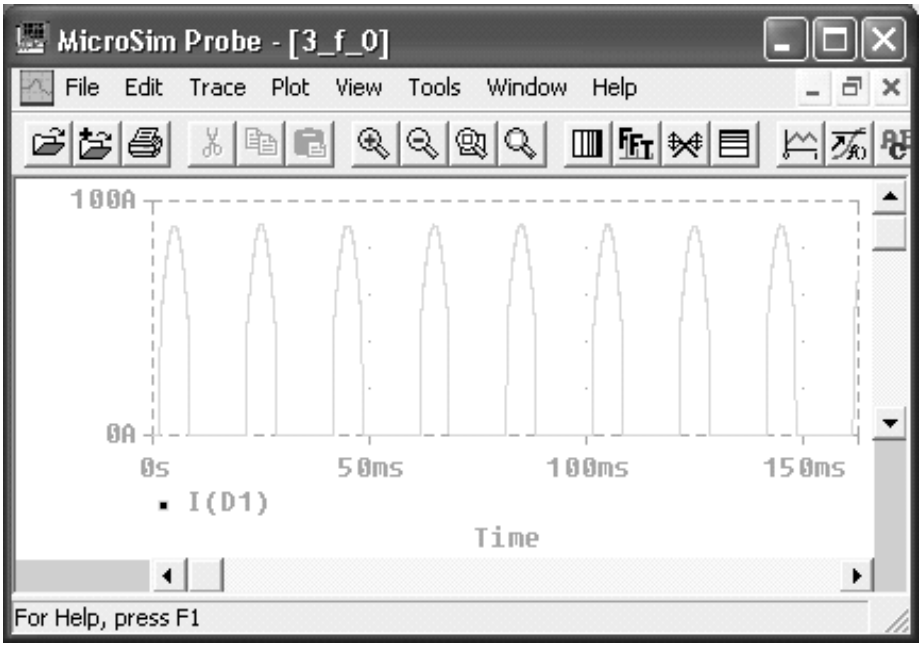


Рисунок 5.19 – Окно программы *Probe*

На окне схемы можно пометить маркером любую цепь или вывод компонента, и в окне программы *Probe* немедленно будет построен соответствующий график. В строке меню есть электронный курсор  , который позволяет при его движении выводить значения всех переменных. В программе Schematics имеется несколько маркеров (напряжения, тока и др.), расположенные в меню Markers. Маркер *Voltage\Level* измеряет напряжение в любой точке схемы относительно заземленного проводника, маркер *VoltageDifferential* измеряет напряжение между двумя точками, маркер *Currentintopin* измеряет ток в участке схемы (этот маркер устанавливается на схеме только в контакте с элементом схемы, не включая провода).

Работа с системой *DesignLab* обычно начинается с создания принципиальной схемы с помощью редактора схем *Schematics*. Он позволяет создавать чертежи принципиальных схем и осуществлять запуск других программ (*PSpice, Probe* и др.). Моделируемая схема может состоять из следующих типовых компонентов: резисторы, конденсаторы, индуктивности, трансформаторы (в том числе с магнитными сердечниками), диоды (включая стабилитроны и варикапы), биполярные, арсенид-галлиевые, полевые, МОП – и биполярные статически индуцированные транзисторы, ключи, управляемые током и напряжением, линии передачи с потерями, аналого-цифровые и цифро-аналоговые преобразователи, цифровые элементы (вентили, триггеры, устройства контроля, запоминающие устройства и программируемые логические матрицы).

Создание схемы начинается с размещения компонентов. Выбор компонента можно осуществить из алфавитного списка в отдельной библиотеке либо путем поиска по ключевым словам по всем компонентам. Наиболее удобным является способ выбора компонентов из соответствующих библиотек. Для этого следует нажать кнопку выбора компонентов  или выполнить команду Draw/ GetNewPart, в раскрывшемся окне PartBrowserBasicнажать кнопку Libraries(библиотеки), в поле Libraryвыбрать библиотеку (рис. 5.20) Analog.slb (аналоговые компоненты), а в поле Part – нужный компонент.

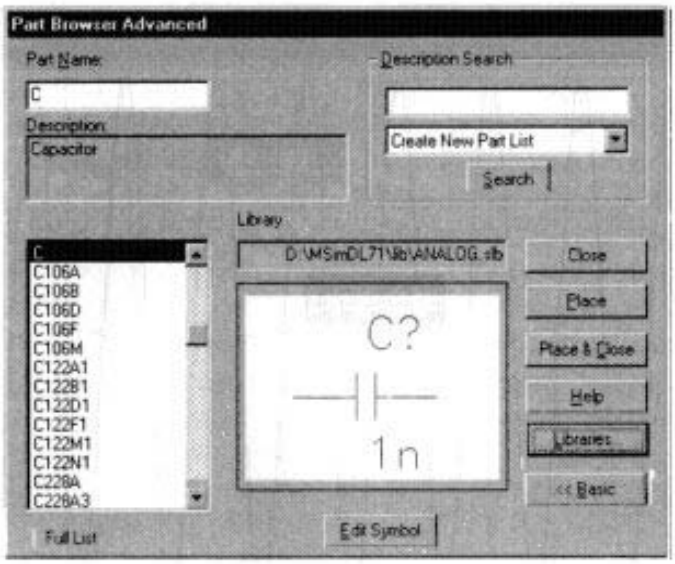


Рисунок 5.20 – Библиотека

Создаем простую схему, выбирая элементы из библиотеки и соединяя их в необходимой последовательности (рис. 5.21).

Для моделирования процесса работы подаем на схему входные воздействия (рис. 5.22) с помощью стимулов (DigStim). Входные воздействия и интервал моделирования схемы выберем такие, чтобы можно было оценить качество работы схемы, адекватность модели (время моделирования 2.2 мкс и входные воздействия заданы программой StimulusEditor).

Расставляем маркеры для снятия параметров работы модели, также с помощью постпроцессора моделирования *Probe* объединяем множества связных сигналов D3,D2,D1,D0 и S1,S0 в шины DBUS и Address для упрощения восприятия результатов моделирования.

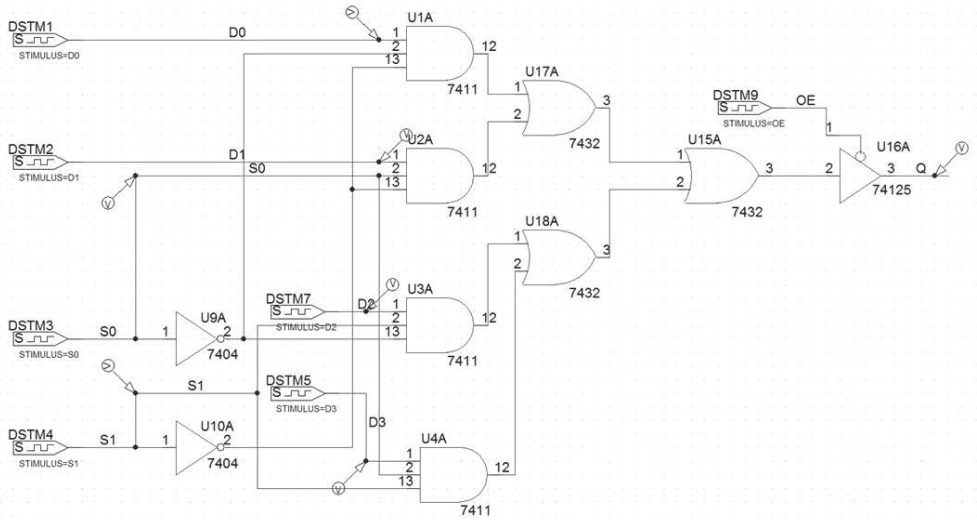


Рисунок 5.21 – Цифровая схема устройства

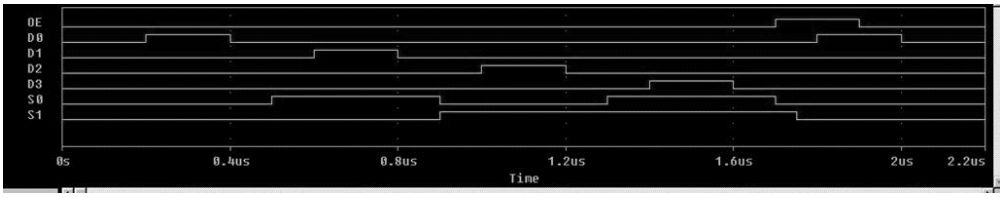


Рисунок 5.22 – Задание входных воздействий на модель

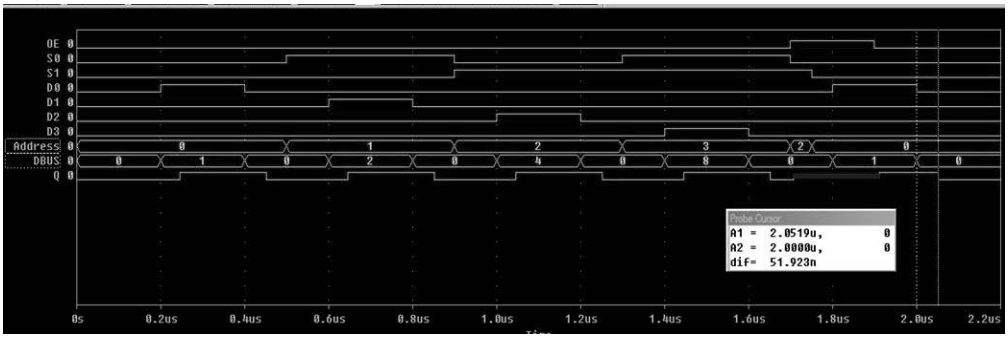


Рисунок 5.23 – Временная диаграмма работы устройства

Из диаграммы видно (рис. 5.23), что модель работает согласно её идее и задержка реакции выходного сигнала на переключение шины данных равна 51.93 нс, аналогично можно измерить другие задержки. Также из моделирования видно, что задержка переключения выходного сигнала из 1 в 0 не равна задержке переключения из 0 в 1

Первый этап проектирования (рис. 5.24) – создание входного описания проекта (EntryDesign).

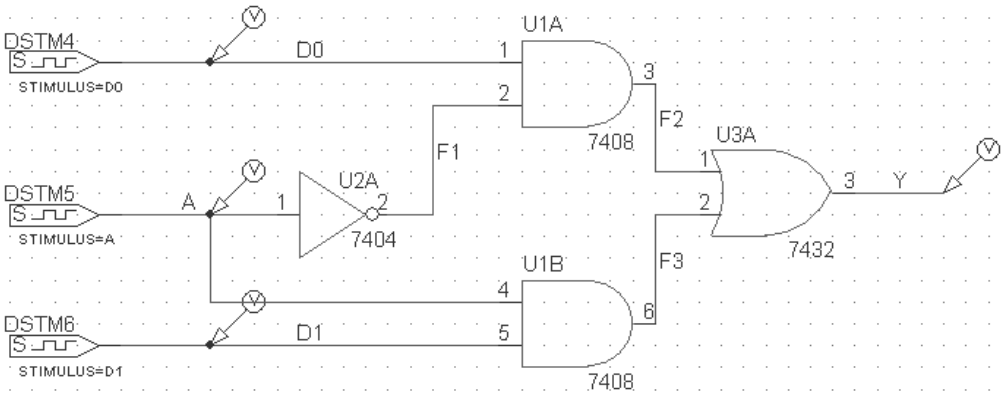


Рисунок 5.24 – Проектирование на корпусных микросхемах

На втором этапе выполняется логическое моделирование (рис. 5.25) (функциональная верификация) с целью проверки правильности работы.

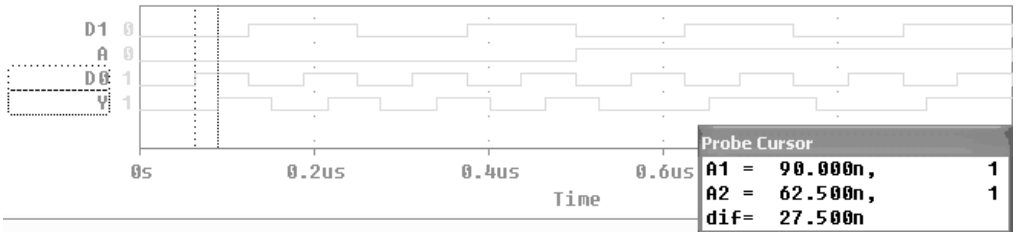


Рисунок 5.25 – Логическое моделирование (функциональная верификация)

На временных диаграммах видна задержка распространения сигнала. В данном случае измерена задержка от входа D0 до выхода Y. Она составляет 27.5 ns.

На третьем этапе верифицированная схема подготавливается для создания печатной платы (рис. 5.26). В частности, из схемы удаляются генераторы входных сигналов (стимулы), и вместо них устанавливаются разъёмы.

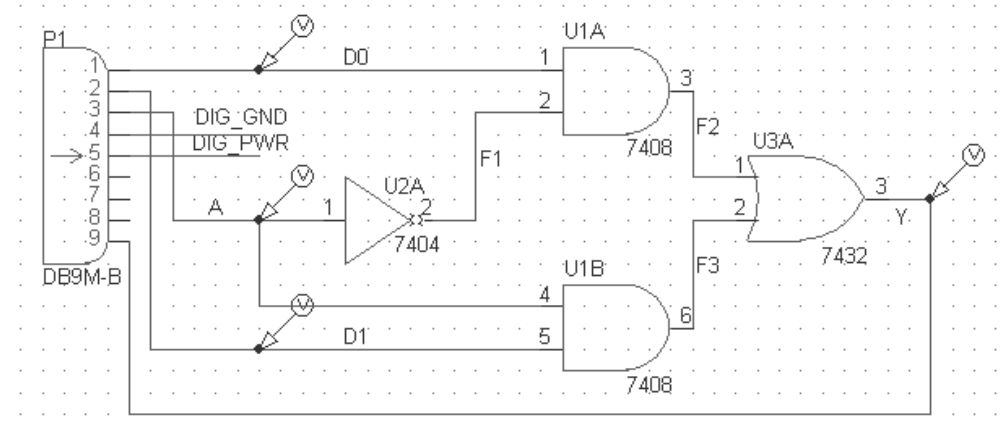


Рисунок 5.26 – Создание печатной платы

На четвёртом этапе в автоматическом режиме выполняется трассировка печатной платы (рис. 5.27).

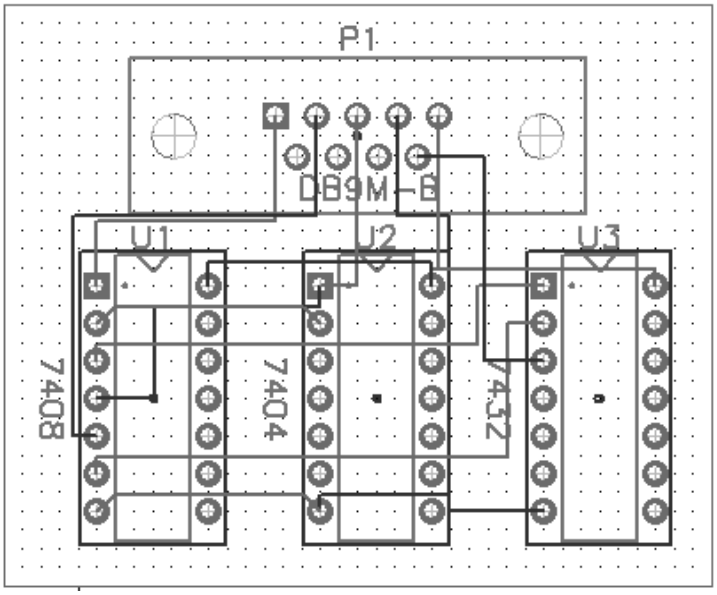


Рисунок 5.27 – Трассировка печатной платы

ВЫВОД

С помощью данного пакета можно исследовать работу схем на разных скоростях, что очень удобно. Программа выдаёт все нарушения в виде сообщений, таким образом можно отслеживать медленные элементы в схеме и улучшать ей качество. В целом продукт очень полезен для моделирования, он обеспечивает сквозное проектирование и удобен на каждом уровне проектирования.

Наряду с развитием цифровых вычислительных машин формировалось направление аналоговых вычислительных машин (АВМ), с помощью которых решались различные физические и математические задачи. АВМ позволяли решать различные виды математических моделей, представленных в виде дифференциальных уравнений с помощью натурного схемотехнического моделирования. Аналоговые ЭВМ в настоящее время не разрабатываются. Однако появились технические информационные системы (компьютерные виртуальные конструкторы), в частности, *Electronics Workbench, Vissim, LabVIEW*, приложение *Simulink* системы *MATLAB* и другие системы, решающие математические задачи с помощью схемотехнического моделирования.

Системы технического моделирования построены по принципу конструктора, по системе блоков. В системах технического моделирования можно решать как математические, так и инженерные задачи. В этих компьютерных системах можно собирать и конструировать виртуально любые электротехнические схемы с использованием компьютерных аналогов электротехнических и измерительных деталей, а также визуальное моделирование и конструирование инженерных, технических имитаторов электронных приборов и логических устройств. Более того, проектированные и созданные виртуальные инженерные и производственные компьютерные объекты и установки можно использовать для натурного эксперимента и производственных испытаний в реальном масштабе времени.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Library Browser?
2. Какое расширение имеют файлы – модели *Simulink*?
3. Как создать новую модель?
4. Как передать результаты моделирования в рабочую область *MATLAB*? В каком виде они передаются?
5. Что определяется в процессе анализа системы?
6. Что определяется в процессе синтеза системы?
7. Чем оценивается эффективность системы?
8. Чем инверсный критерий эффективности отличается от прямого?
9. Что понимается под оптимальной системой?
10. Свойства, присущие сложной системе, и их краткая характеристика.
11. Что такое техническое моделирование?
12. В каких случаях моделирование оправдано и необходимо?
13. Чем отличается реализация функциональной организации системы от структурной?
14. Что определяется в процессе анализа системы?
15. Какой метод исследования систем является наиболее точным?
16. Что такое модель и моделирование?
17. Для чего предназначена программа *VisSim*?
18. В чем заключаются принципы функционирования программы *VisSim*?
19. Что такое «техническое моделирование»?
20. Классификация технических систем в моделировании.
21. Поясните различие между геометрическими и физическими моделями изучаемых систем.
22. Назначение и особенности программной среды *Labview*.
23. Порядок программирования простых виртуальных приборов.
24. Чем оценивается эффективность системы?
25. Чем инверсный критерий эффективности отличается от прямого?
26. Что понимается под оптимальной системой?
27. Свойства, присущие сложной системе, и их краткая характеристика.
28. Что такое техническое моделирование?
29. Что такое «*Multisim*?
30. Зачем нужно заземление?
31. Что такое цепь?

6. **СИСТЕМЫ ГРАФИЧЕСКОГО МОДЕЛИРОВАНИЯ**

Проектно-графическое моделирование — это процесс отражения средствами графики пространственных характеристик идеального образа проектируемого изделия или комплекса, способ мыслительной и практической деятельности дизайнера. Результатом проектно-графического моделирования является модель – визуализация идеальной воображаемой, знаковой или материально реализованной системы создаваемой в целях исследования объекта или представления проектной идеи. Самой продуктивной и активной в сознании проектировщика является модель будущего объекта, поскольку содержит в себе синтезированные представления о связях человека с окружающим миром. Изображение, передающее информацию о виде предмета и как бы заменяющее при этом сам предмет, является его визуальной моделью. Такие визуальные модели по своей информативной насыщенности могут быть различными: от условного изображения, зафиксировавшего только один главный визуальный признак, до полного воспроизведения всех визуальных свойств объекта.

Наблюдая за стремительным развитием компьютерной техники, мы одновременно становимся свидетелями того, насколько быстро меняются системы моделирования – системы для создания графических образов. Они становятся все более интуитивными и понятными в работе, все более усиливается впечатление, будто мы работаем не с точками, кривыми и цветом, а с реальными объектами, которые так же ощутимы, как и окружающие нас предметы. Первым шагом вперед был переход от двухмерных систем моделирования к трехмерным.

Трёхмерная графика – раздел компьютерной графики, совокупности приемов и инструментов (как программных, так и аппаратных), предназначенных для изображения объёмных объектов.

3D моделирование – это процесс создания трехмерной модели объекта. Задача 3D моделирования – разработать визуальный объемный образ желаемого объекта. С помощью трехмерной графики можно и создать точную копию конкретного предмета, и разработать новое, даже нереальное представление до сего момента не- существовавшего объекта.

6.1. **Конструирование объектов с использованием пакета** Autodesk 3ds Max (**ранее** 3D Studio MAX)

Autodesk 3ds Max (ранее 3D Studio MAX) – полнофункциональная профессиональная программная система для создания и редактирования трёхмерной графики и анимации, доработанная компанией Autodesk. Содержит самые современные средства для художников и специалистов в области мультимедиа. Работает в операционных системах Microsoft Windows и Windows NT (как в 32-битных, так и в 64-битных). В апреле 2013 года выпущена шестнадцатая версия этого продукта под названием «Autodesk 3ds Max 2014». Написана на C# (WPF), также использует библиотеку DeveloperExpress (DevExpress). *Autodesk 3ds Max* доступен в двух лицензионных версиях: студенческая – бесплатная (требуется регистрация на сайте Autodesk), которая предоставляет полную версию программы (однако, её нельзя использовать с целью получения прибыли). Первая версия пакета под названием 3D Studio DOS была выпущена в 1990 году. Разработками пакета занималась независимая студия YostGroup, созданная программистом Гари Йостом; Autodesk на первых порах занимался только изданием пакета. Первые четыре релиза носили наименование 3D Studio DOS (1990-1994 годы). Затем пакет был переписан заново под Windows NT и переименован в 3D Studio MAX (1996-1999 годы). Нумерация версий началась заново. В 2000-2004 годах пакет выпускается под маркой Discreet 3dsmax, а с 2005 года – *Autodesk 3ds MAX*. Актуальная версия носит название *Autodesk 3ds MAX 2014* (индекс 16.0).

Программа обладает интерактивным объектно-ориентированным интерфейсом, реализует расширенные возможности создания и управления анимацией, хранит историю жизни каждого

объекта, предоставляет возможности для создания разнообразных световых эффектов и имеет открытую архитектуру, что позволяет расширять возможности приложения за счет подключаемых плагинов. *3D Studio MAX* обладает всеми необходимыми средствами для создания игровых миров и анимационных роликов и потому используется большинством разработчиков компьютерных игр и незаменим в компьютерной мультипликации и художественной анимации. Дизайнерам и инженерам *3D Studio MAX* предоставляет средства фото реалистической визуализации для анализа разрабатываемого проекта, проведения презентаций и создания маркетинговых материалов. Широко применяется он в архитектурном проектировании для создания дизайна интерьеров. Давно оценили данное приложение и специалисты по телевизионным заставкам, клипам и спецэффектам в кино, пакет широко применяется при подготовке рекламных и научно-популярных роликов для телевидения.

3D StudioMax может использоваться:

- в архитектурном проектировании (рис. 6.1);

Рисунок 6.1

- в подготовке рекламных и научно-популярных роликов для телевидения (рис. 6.2);

Рисунок 6.2

- в компьютерной мультипликации и художественной анимации (рис. 6.3);

Рисунок 6.3

- в компьютерных играх (рис. 6.4);

Рисунок 6.4

- в компьютерной графике и Web-дизайне (рис. 6.5).

Рисунок 6.5

В отличие от программ 2D графики, в *3ds Max* мы имеем дело с трехмерными объектами, которые обладают такими характеристиками, как ширина, длина и высота. Таким образом, совершенно очевидно, что для достоверного отображения таких объектов необходимо использовать три различных его вида. *3ds Max* по умолчанию использует вид сверху (Top), слева (Left) и спереди (Front). Также имеется еще одна проекция – перспектива (Perspective), которая используется скорее для контроля за сценами, так как работать в ней довольно сложно (см. рис. 6.6). В любом случае, отображения в окнах проекции можно свободно переключать, а так же увеличивать в четыре раза и уменьшать до исходного размера, о чем будет сказано ниже.

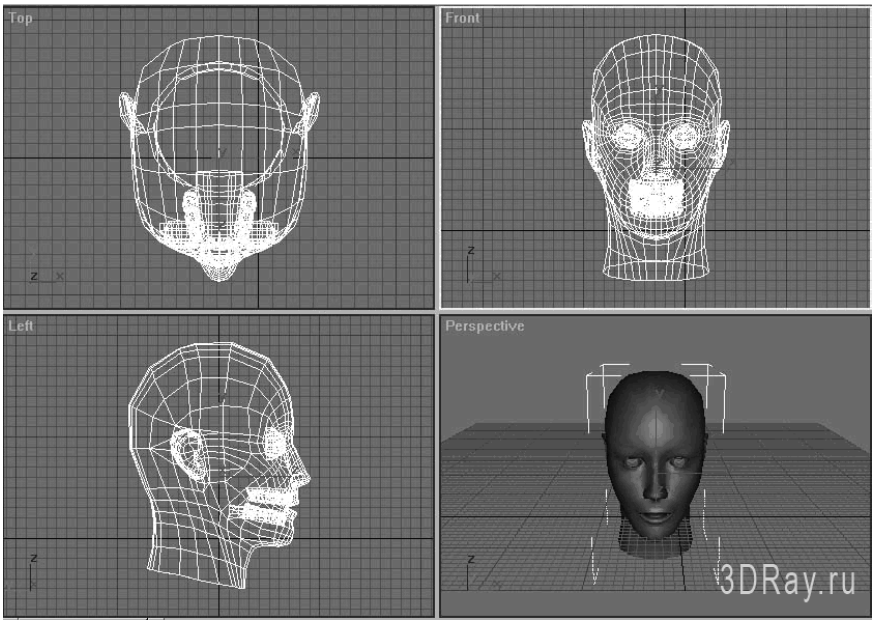
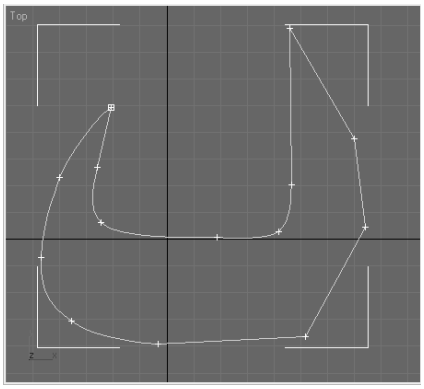


Рисунок 6.6

Рассмотрим программу на примере консервного ножа:

При моделировании в *3ds Max* мы создаем с помощью линии контуры лезвия и ручки (рис. 6.7):

С помощью команды Smooth делаем их вид более сглаженным после настройки и вращения путем выдавливания добиваемся 3д эффекта.



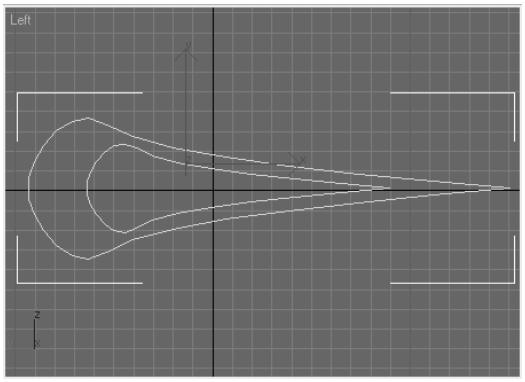
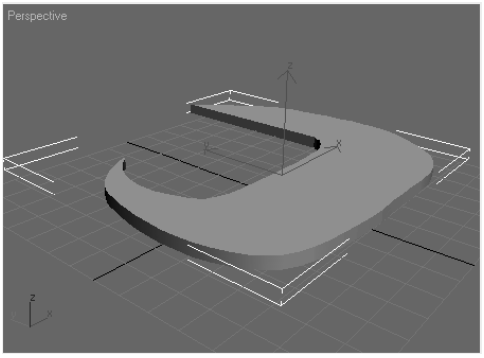


Рисунок 6.7

Соединяем компоненты и для естественности дополняем объект элементами. После добавления текстуры объект приобретает законченный вид (рис. 6.8, 6.9).



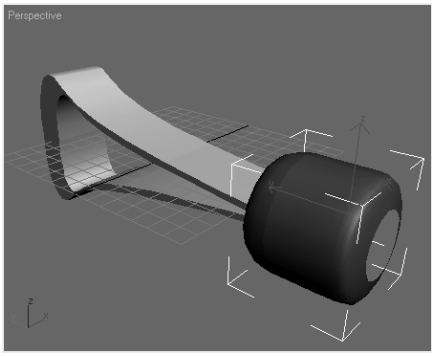


Рисунок 6.8

Рисунок 6.9

Отдельный набор текстурных карт, организующих наложения и объединения других текстур в сложные составные карты, который выделен в группу Compositors (Многокомпонентные). Их при-

менение необходимо в материалах с усложненной иерархией связей и избавляет от необходимости дополнительной обработки исходных файлов в растровых редакторах типа Adobe Photoshop.

Построим вентилятор и заставим его лопасти вращаться.

1. Щелкаем левой кнопкой мыши на кнопке Geometry и выбираем в списке разновидностей объектов — Standard Primitives.

2. Щелкаем на кнопке Box.

3. Переходим в окно Top (Вид сверху) и нарисуем там параллелепипед с длиной, например 3м, шириной 2м и высотой 0.2м.

4 Щелкаем на кнопке Cilynder.

5. В окне Top (Вид сверху) нарисуем цилиндр с радиусом 0,3м и высотой 2,5м.

6. Теперь выберем в списке разновидностей объектов — Extended Primitives, . нажмем кнопку Chamfer Box (Параллелепипед с фаской).

7. Размер фаски выберите по своему усмотрению.

9. Группируем.

10. Теперь нарисуем сам винт с лопастями.

11.Чтобы заставить лопасти винта вращаться, нажмем на большую кнопку AutoKey (Автоключ или Анимация) в нижней части экрана программы. Она приобретет красный цвет. Затем передвинем ползунок анимации до упора, чтобы на нем было написано (100/100). Далее еще раз нажмём кнопку AutoKey (Анимация), чтобы она стала неактивной. Все, анимация готова. Просмотреть ее вы можете либо двигая ползунок, либо нажав кнопку Play, которую, надеюсь, вы уже обнаружили возле кнопки AutoKey (рис. 6.10).

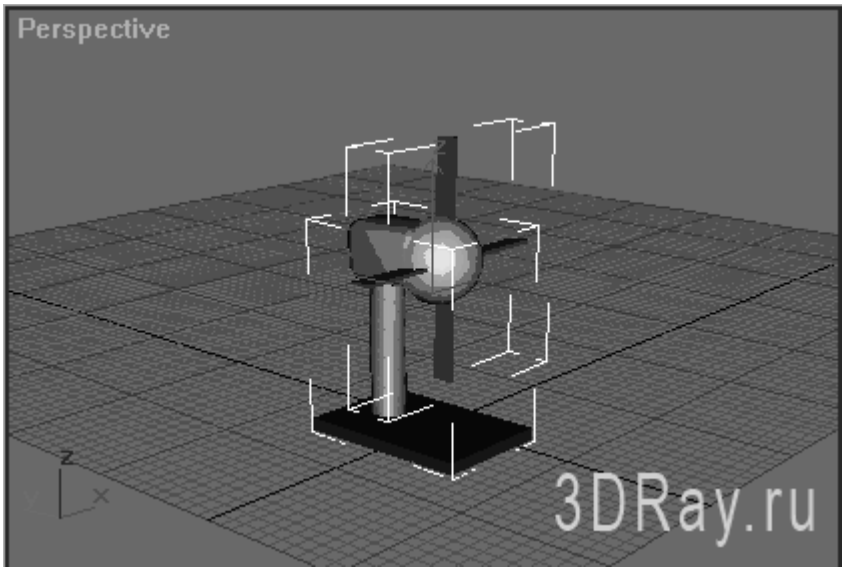


Рисунок 6.10

6.2. **Конструирование объектов с использованием пакета** Maya

Maya можно назвать средой разработки инструментов для производства трехмерной компьютерной графики. Не фиксированным набором средств, а именно средой, в которой наряду с уже имеющимися огромными возможностями можно создавать новые инструменты и разрабатывать

новые методы для получения необходимых эффектов и результатов. *MAYA* не похожа на конкретный механизм с четко определенным набором функций – это скорее конструктор, позволяющий комбинировать уже имеющиеся работающие блоки в нужной последовательности, а при необходимости легко создавать новые блоки. *MAYA* является, действительно, современным набором инструментов, вобравшим в себя последние идеи и технологии компьютерной графики. Средства моделирования включают в себя несколько технологий. Во-первых, это довольно удобный набор инструментов для полигонального моделирования. Во-вторых, операции NURBS-моделирования, позволяющие работать со сплайновыми моделями действительно быстро.

Сохранение же истории моделирования позволяет экономить массу времени и не переделывать заново множество операций. Поверхности разбиения (Subdivision Surfaces) являются симбиозом сплайнов и полигонов.

В *Maya* существует три типа поверхностей для моделирования:

- Polygons;
- NURBS;
- Subdivision surfaces.

На основании этих типов соответственно выделяют и методы моделирования.

Каждый из методов имеет свои характеристики и плюсы.

Полигон – значит многоугольник. Может состоять минимум из трех сторон и более. Вершины (Vertex) соединяются прямыми линиями (ребрами – edges), внутренняя область полигона называется гранью (face) (рис.6.11).

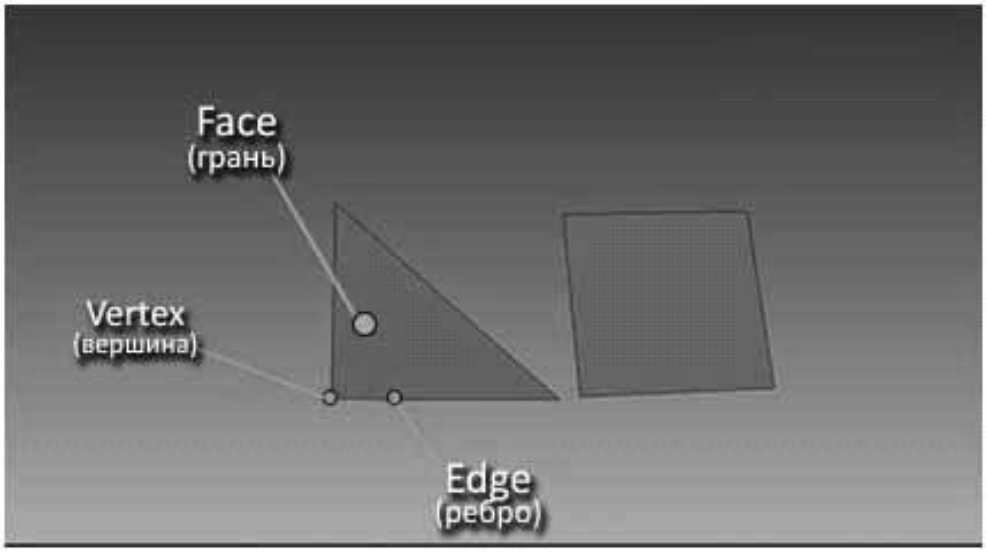


Рисунок 6.11 – Трех- и четырехсторонние полигоны

Полигональные поверхности (Polygon surfaces) - это сеть из трех- или более сторонних поверхностей, называемых гранями, которые соединяются вместе, создавая полигональную сетку. Полигональная сетка состоит из вершин, граней и ребер.

Полигональное моделирование очень простое в использовании и понимании. Немного освоив полигональное моделирование, легко с одного объекта, к примеру, куба, получить сложный детализированный объект.

Каркасные линии сетки представляют собой ребра каждой грани. Области, ограниченные ребрами, - это грани. Точки, в которых ребра пересекаются друг с другом, называются вершинами.

Когда полигональная сетка визуализируется, ее ребра могут быть установлены жесткими или сглаженными. В результате, полигоны могут с легкостью представить как плоскую, так и 3D форму.

Существуют разные техники по созданию полигональных моделей в *MAYA*:

Примитивы (рис. 6.12) – это трехмерные геометрические формы, которые можно создать в *MAYA*. В число примитивов входят, например, сфера, куб, цилиндр, конус, плоскость и многие другие.

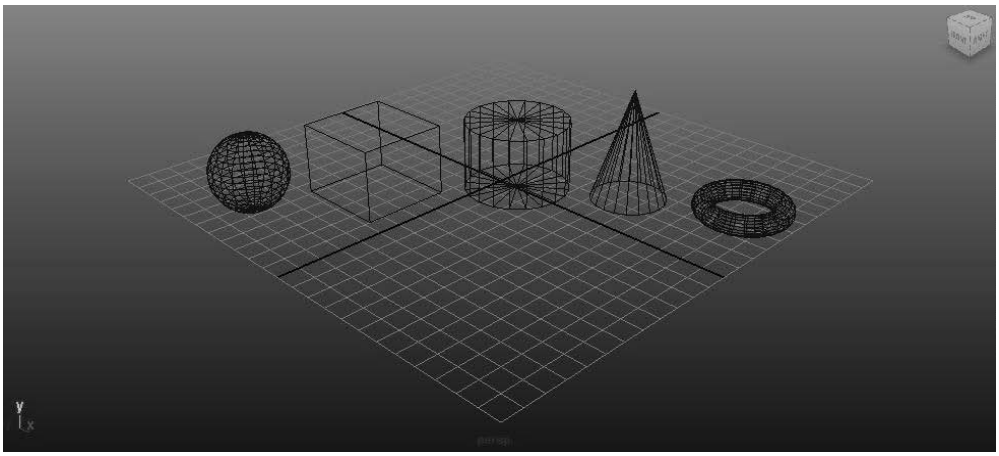


Рисунок 6.12 – Полигональные примитивы

Можно изменить базовые атрибуты примитива, чтобы сделать его более сложным. Можно также разрезать, экструдировать, сливать или удалять разные компоненты примитива, чтобы изменить его форму. Многие 3D художники используют примитивы в качестве начальной точки для создания моделей. Такая техника называется моделированием из примитива.

Отдельные полигоны могут быть созданы с помощью средства Create Polygon Tool. Этот инструмент позволяет размещать в сцене вершины, определяющие форму полигональной грани. Возможно также разрезать или экструдировать полигональную грань, чтобы добавить новые грани к уже созданной. Такая техника обычно используется, если нужно наиболее точно построить модель по заданному контуру.

Полигональные поверхности имеют широкий спектр применения для многих 3D приложений, включая интерактивные игры и приложения, веб-разработки.

Термин «сплайны» означает параметрические кривые и поверхности, которые бывают разных типов. Самый сложный и гибкий класс называется NURBS (non-uniform rational b-splines).

Слово «неравномерные» (non-uniform) означает возможность неравномерной параметризации вдоль протяженности поверхности, хотя в основном работа ведется с равномерными (uniform) сплайнами.

«Рациональные» (rational) - означает, что контрольные точки могут иметь разные веса и притягивать к себе поверхность по-разному, хотя опять же, по умолчанию работа в *MAYA* ведется с нерациональными поверхностями и кривыми, веса точек у которых одинаковы.

Кривые и патчи Безье – более примитивный подкласс NURBS, в котором возможно редактировать объекты только в конкретных узловых (knot) точках. В MAYA этот, более простой тип сплайнов не используется для моделирования. Таким образом, термины «сплайновые объекты» и «NURBS-объекты» являются синонимами.

Термин «патчи» применительно к *MAYA* имеет двоякое значение. Во-первых, это просто NURBS-поверхности, как правило, с гладко сшитыми краями, лоскутно составляющие модель. Отсюда и термин «патчевое моделирование». Во-вторых, патчами формально называются компоненты сплайновых поверхностей, представляющие собой ячейки в сетке изопарм.

Основой NURBS поверхности является кривая (рис. 6.13). Поверхность создается пересечением кривых. Кривые помогают создавать и модифицировать поверхности (рис. 6.14). Таким образом, создание и редактирование кривых есть важная часть NURBS моделирования. Невозможно визуализировать кривые, так что они применяются только для создания и редактирования поверхностей.

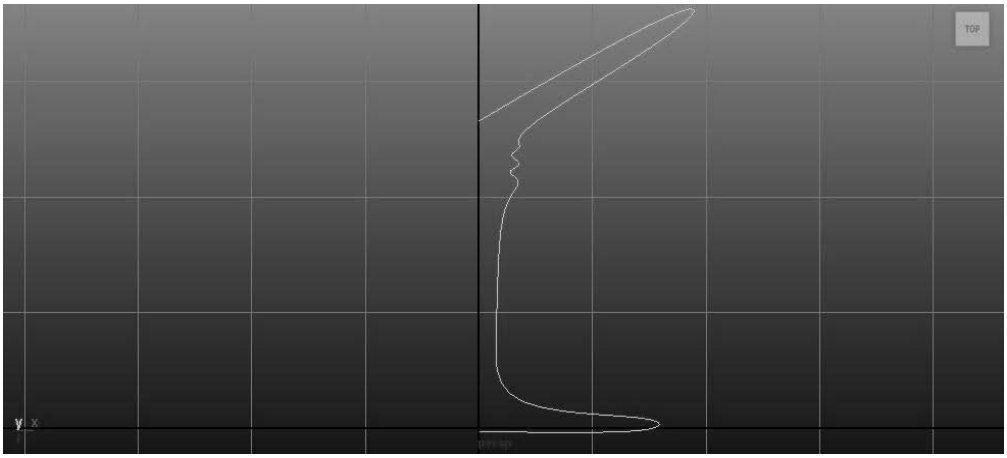


Рисунок 6.13 – NURBS кривая

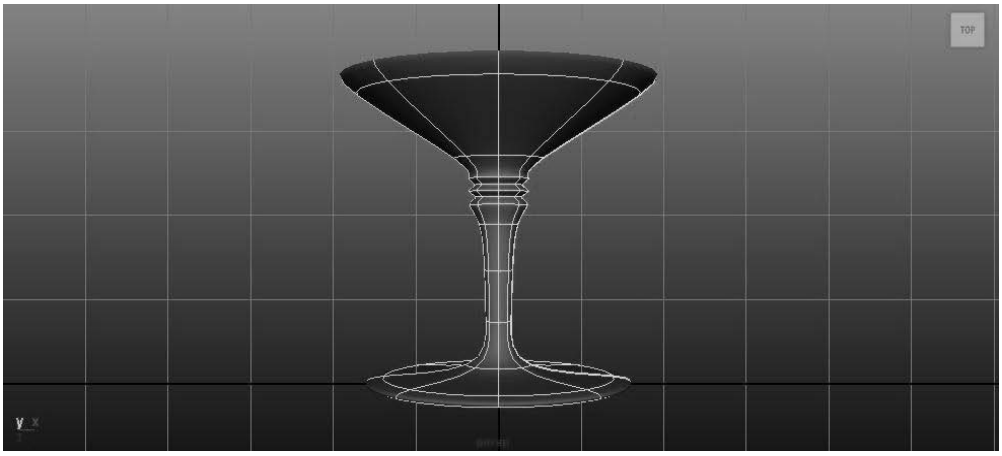


Рисунок 6.14 – NURBS объект

NURBS кривые и поверхности имеют множество применений и предпочтительных типов поверхностей для индустриальных и автомобильных дизайнеров, где необходимы сглаженные формы с минимальным количеством данных для определения конкретных форм. NURBS кривые идеальны для определения сглаженных путей движения для анимированных объектов. NURBS поверхности могут быть смоделированы и затем сконвертированы в полигональную сетку.

Subdivision surface (поверхности с разбиением) – это гладкая поверхность, которая генерируется по произвольной полигональной сетке. SubDs поверхности содержат лучшие характеристики NURBS и полигональных объектов.

Термин «сабдивы» окончательно прижился для обозначения subdivision surfaces, или «поверхностей разбиения» (это, правда, довольно неуклюжий термин: лучше уж было бы их назвать «поверхности сглаживания»).

Представим себе полигональный кубик (созданный в *MAYA*). Применим к нему полигональную операцию Smooth (сглаживание). И где-то после двенадцатой попытки, которая займет неопределенное время (весьма существенное), мы увидим на экране гладкую сферу с некоторым количеством вершин. А теперь представим, что будет после применения беконечного количества операций smooth. Результатом и будет subdivision surface, или сабдив.

Математические методы, позволяющие быстро вычислять subdivision surfaces, появились лишь недавно, и это радикально изменило подход к трехмерному моделированию многих типов объектов.

Дело в том, что subdivision surfaces обладают такой же степенью гладкости, как и NURBS. Любой сабдив может быть представлен как набор гладко стыкующихся кубических NURBS-патчей. Но при этом он остается одной поверхностью!

Следовательно, subdivision surfaces унаследовали все преимущества сплайнов плюс одно маленькое, скромное обаяние полигонов. Поэтому моделирование в сабдивах ведется привычными полигональными методами, а результат отображается в виде гладкой поверхности а-ля NURBS:

- как и NURBS эти поверхности являются сглаженными;
- как и полигональные поверхности - имеют произвольную топологию;
- подобно полигональным поверхностям, возможно работать с одним объектом, а не собирать воедино части, как это происходит с NURBS объектами;
- subdivision surfaces поверхности непрерывны, соответственно не будет проблем со швами во время анимации. Такие проблемы возникают с NURBS объектами.

Основным преимуществом Subdivision Surfaces по сравнению с NURBS- поверхностью является способность принимать произвольную топологическую форму, при этом поверхность остается единой и имеет сравнительно небольшое количество базовых контрольных вершин. Это достигается благодаря тому, что подразбиваемые поверхности за счет своей внутренней иерархической структуры позволяют динамически увеличивать и уменьшать детализацию моделируемых поверхностей лишь на тех участках, где это необходимо. Таким образом, более сложные в моделировании области с множеством мелких деталей могут иметь больше контрольных вершин, а значит, более тщательно прорабатываться, в то время как однородные плоские области будут ограничены гораздо меньшим числом вершин. Для сравнения стоит напомнить, что для увеличения детализации сплайновых поверхностей приходится вставлять дополнительные изопармы (например, в области глаз при моделировании лица), которые проходят через всю поверхность. Это приводит к добавлению большого количества контрольных вершин не только в тех областях, где это необходимо, но и в других, где это совершенно не требуется, – в итоге модель усложняется. В Subdiv-моделях увеличение детализации производится не путем добавления изопарм, а за счет появления дополнительных управляющих вершин, которые добавляются лишь внутри указанной области.

Вторым важным преимуществом Subdiv-поверхностей является то, что они позволяют создавать сложные модели в виде полностью непрерывных поверхностей, без швов (одна модель из одной поверхности), в то время как при NURBS-моделировании модель сшивается из отдельных NURBS-поверхностей, что требует немало времени и приводит к ряду проблем, особенно при анимации.

SubDs объекты как бы помещаются в полигональный каркас. При модифицировании каркаса происходит грубое изменение SubDs поверхности. Чтобы увидеть базовый каркас объекта, при выбранном объекте нажмите ПКМ и выберите команду Polygon. SubDs поверхность поместится в полигональную коробку, теперь вы сможете работать точно так же ,как и с полигонами, не заботясь о сглаженности поверхности.

Теоретически можно выделить такие методы построения моделей с применением подразбиваемых поверхностей. Самый простой – воспользоваться Subdiv-примитивами: Sphere

(Сфера), Cube (Куб), Cylinder (Цилиндр), Cone (Конус), Plane (Плоскость) и Torus (Тор). Как и другие типы примитивов, Subdiv-примитивы (рис. 6.15) обычно используются в качестве основы для формирования более сложных моделей и создаются командой Create=>Subdiv Primitives либо выбором нужного примитива на вкладке Subdivs (Поверхности) панели Shelf.

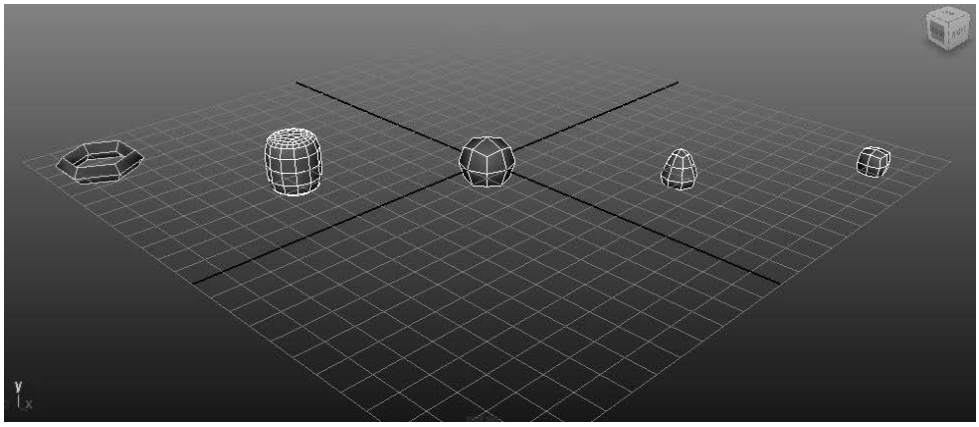


Рисунок 6.15 – Subdiv примитивы

При создании поверхностей с иерархическим разбиением (рис.6.16) необходимо выполнять следующие правила:

- начинать построение Subdiv-модели лучше с простой базовой сетки, так как получить грубую аппроксимацию модели на базе полигональной поверхности с небольшим числом полигонов гораздо быстрее, нежели добиться того же результата на сложной полигональной сетке. Как правило, данное условие выполняется автоматически при конвертировании в Subdiv-модель несглаженных полигональных поверхностей или в случае создания ее на базе Subdiv-примитивов. Однако бывают ситуации, когда требуется применить Subdiv-моделирование для уже существующей довольно сложной модели (например, представленной NURBS-поверхностями), тогда перед созданием поверхности с иерархическим разбиением исходную NURBS-модель по возможности упрощают;
- создавая исходную модель, которую в дальнейшем предполагается переконвертировать в подразбиваемую поверхность, желательно использовать в ней только четырехугольные полигоны. Дело в том, что в противном случае на нулевом уровне Subdiv-поверхности появятся экстраординарные вершины, в которых сходятся не четыре ребра, а либо большее, либо меньшее их количество, что чревато появлением на поверхности нежелательного рельефа;
- необходимо контролировать положение ребер и вершин в исходной геометрии, стараясь располагать их на участках предполагаемого усложнения геометрии объекта, что существенно облегчит дальнейшую работу с моделью. Например, наличие ребер желательно там, где предполагается создавать глаза и рот персонажа, складки на одежде и т.п.

MAYA позволяет конвертировать один вид поверхности в другой, т.е. NURBS можно преобразовать в полигональные и Subdivision поверхности. Точно также полигональные и Subdivision — поверхности можно преобразовать в другой вид поверхности. Есть одно исключение: полигональные поверхности нельзя конвертировать в NURBS объекты. Конвертировать поверхности можно в меню Modify>Convert, там найдете полный перечень конвертаций.

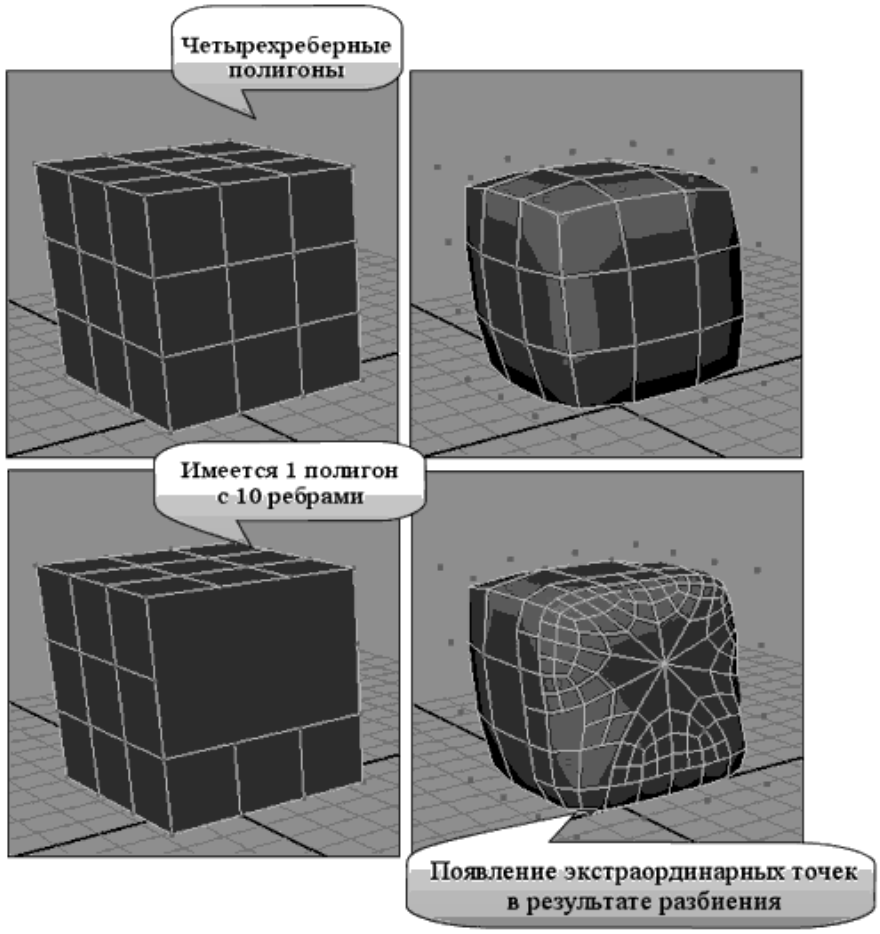


Рисунок 6.16 – Результат Subdiv-конвертирования модели с четырехугольными (слева) и различными по числу ребер полигонами

Изначально проектируемая для индустрии развлечений MAYA неожиданно распространилась как в смежные, так и в довольно отдаленные области. Никого не удивишь тем, что MAYA используется в кинопроизводстве, рекламном бизнесе или в игровой индустрии. Благодаря открытому формату данных MAYA нашла интенсивное применение в сфере научной визуализации, когда огромные массивы уже полученных данных чрезвычайно гибко и интерактивно представляются на экране с помощью разнообразных инструментов *MAYA*.

Промышленные дизайнеры сразу оценили возможности быстрой и качественной визуализации, доступные в *MAYA*.

Возможности выразительного органического моделирования и наглядной визуализации сделали *MAYA* привлекательной и для медицины.

Список можно продолжать, ибо благодаря открытости *MAYA* области ее применения весьма широки.

Кстати, банальный факт использования *MAYA* в индустрии компьютерных игр можно дополнить нетривиальным вариантом использования *MAYA* в качестве интереснейшей, красивой, (и трехмерной!) компьютерной игры с бесконечным количеством уровней.

6.3. **Конструирование объектов с использованием пакета** SolidWorks

SolidWorks – система автоматизированного проектирования, инженерного анализа и подготовки производства изделий любой сложности и назначения. *SolidWorks* является ядром интегрированного комплекса автоматизации предприятия, с помощью которого осуществляется поддержка жизненного цикла изделия в соответствии с концепцией CALS-технологий, включая двунаправленный обмен данными с другими Windows-приложениями и создание интерактивной документации.

Разработки *SolidWorks Corp.* характеризуются высокими показателями качества, надежности и производительности, что в сочетании с квалифицированной поддержкой делает *SolidWorks* лучшим решением для промышленности.

Комплексные решения *SolidWorks* базируются на передовых технологиях гибридного параметрического моделирования и широком спектре специализированных модулей. Программное обеспечение функционирует на платформе Windows XP, выполнено на русском языке, поддерживает ГОСТ и ЕСКД.

Самое главное, что даёт конструктору *SolidWorks,* – это возможность работать так, как он привык, не подстраиваясь под особенности используемой компьютерной системы. Процесс моделирования начинается с выбора конструктивной плоскости, в которой будет строиться двухмерный эскиз. Впоследствии этот эскиз можно тем или иным способом легко преобразовать в твёрдое тело. При создании эскиза доступен полный набор геометрических построений и операций редактирования. Нет никакой необходимости сразу точно выдерживать требуемые размеры, достаточно примерно соблюдать конфигурацию эскиза. Позже, если потребуется, конструктор может изменить значение любого размера и наложить связи, ограничивающие взаимное расположение отрезков, дуг, окружностей и т.п. Эскиз конструктивного элемента может быть легко отредактирован в любой момент работы над моделью.

Пользователю предоставляются несколько различных средств создания объёмных моделей. Основными формообразующими операциями в SolidWorks являются команды добавления и снятия материала. Система позволяет выдавливать контур с различными конечными условиями, в том числе на заданную длину или до указанной поверхности, а также вращать контур вокруг заданной оси. Возможно создание тела по заданным контурам с использованием нескольких образующих кривых (так называемая операция лофтинга) и выдавливанием контура вдоль заданной траектории. Кроме того, в SolidWorks необычайно легко строятся литейные уклоны на выбранных гранях модели, полости в твёрдых телах с заданием различных толщин для различных граней, скругления постоянного и переменного радиуса, фаски и отверстия сложной формы. При этом система позволяет отредактировать в любой момент времени однажды построенный элемент твердотельный модели.

Важной характеристикой системы является возможность получения развёрток для спроектированных деталей из листового материала. При необходимости в модель, находящуюся в развёрнутом состоянии, могут быть добавлены новые места сгиба и различные конструктивные элементы, которые по каким-либо причинам нельзя было создать раньше.

При проектировании деталей, изготовляемых литьём, очень полезной оказывается возможность создания разъёмных литейных форм. Если для работы необходимо использовать какие-либо часто повторяющиеся конструктивные элементы, на помощь приходит способность системы сохранять примитивы в виде библиотечных элементов.

Кроме проектирования твёрдотельных моделей, *SolidWorks* поддерживает и возможность поверхностного представления объектов. При работе с поверхностями используются те же основные

способы, что и при работе с твёрдыми телами. Возможно построение поверхностей, эквидистант, а также импорт поверхностей из других систем с использованием формата IGES.

Значительно упрощают работу многочисленные сервисные возможности, такие как копирование выбранных конструктивных элементов по линии или по кругу, зеркальное отображение как указанных примитивов или модели.

При редактировании конструктор может возвратить модель в состояние, предшествовавшее созданию выбранного элемента. Это может потребоваться для выполнения каких-либо действий, невозможных в текущий момент.

SolidWorks содержит высокоэффективные средства твердотельного моделирования, основывающиеся на постепенном добавлении или вычитании базовых конструктивных тел. Эскиз для получения базового тела может быть построен на произвольной рабочей плоскости.

Типовые инструменты для получения базовых тел позволяют выполнить:

- выдавливание заданного контура с возможностью указания угла наклона образующей;
- вращение контура вокруг оси;
- создание твёрдого тела, ограничиваемого поверхностью перехода между заданными контурами;
- выдавливание контура вдоль заданной кривой;
- построение фасок и скруглений различного вида;
- построение уклонов;
- создание различного типа отверстий;
- получение развёртки тел равномерной толщины.

Основные методы создания твёрдого тела сочетают в себе также возможность комбинации всех перечисленных способов как при добавлении материала, так и при его снятии. Естественный порядок работы конструктора без труда позволяет создавать сложные твёрдотельные модели, состоящие из сотен конструктивных элементов. При необходимости во время работы возможно введение вспомогательных плоскостей и осей для использования в дальнейших построениях.

Параметры всех созданных конструктивных элементов доступны для изменения, так что в любой момент работы можно изменить произвольный параметр эскиза или базового тела и выполнить затем полную перестройку модели.

Кроме создания твёрдых тел, в *SolidWorks* существует возможность построения различных поверхностей, которые могут быть использованы как для вспомогательных построений, так и самостоятельно. Поверхности могут быть импортированы из любой внешней системы или построены теми же способами, что и твёрдые тела (выдавливание, вращение, переход между контурами и т.п.). Допускается получение слепка любой из поверхностей уже построенного твердого тела.

Режимы визуализации полученной модели позволяют просматривать ее каркасное или реалистичное изображение. Для повышения качества тонированных изображений могут быть изменены физические характеристики поверхности детали (текстуры) и назначены дополнительные источники света.

Интерфейс программы достаточно прост в обращении. Все инструменты работы в программе разбиты на группы команд – панели инструментов. На рис. 6.17 представлен общий вид интерфейса *SolidWorks*.

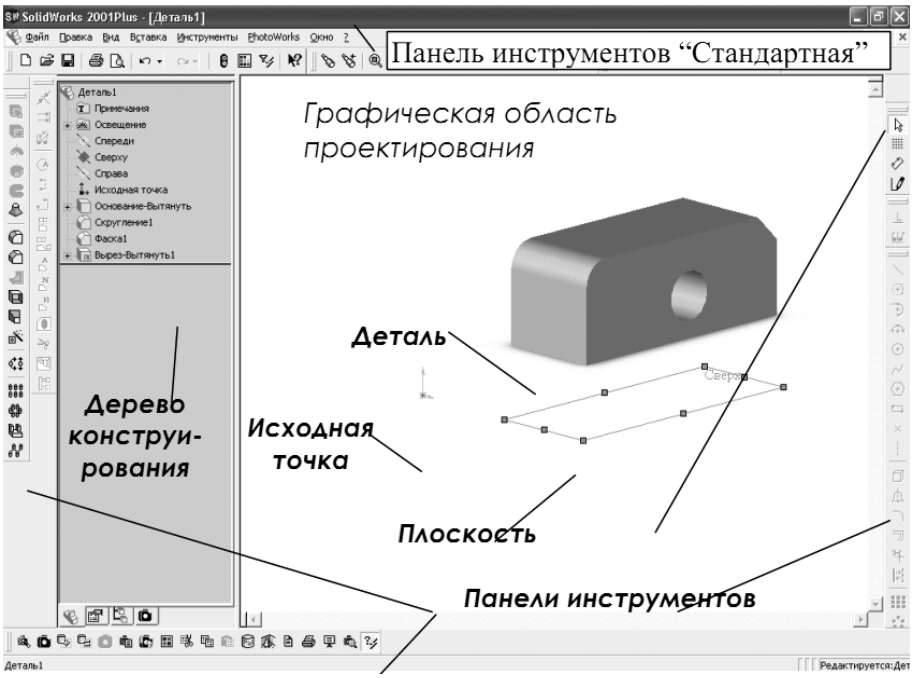


Рисунок 6.17 – Интерфейс программы

*6.3.1. **Принципы создания деталей***

Деталь в программе создают из элементов различных типов (вырезов, отверстий, скруглений, фасок и т.д.) путем их объединения или вычитания. На рис. 6.18 представлены этапы создания детали в *SolidWorks*.

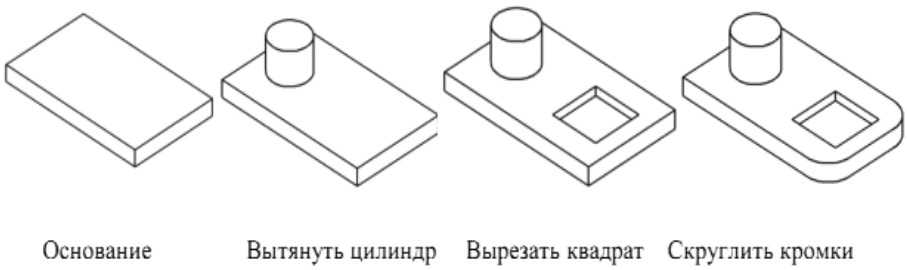


Рисунок 6.18 – Этапы создания детали

Объединение и вычитание элементов различных типов в детали производят на основе принципов объединения и вычитания множеств. На рис. 6.19, 6.20, 6.21 показаны принципы объединения и вычитания элементов детали.

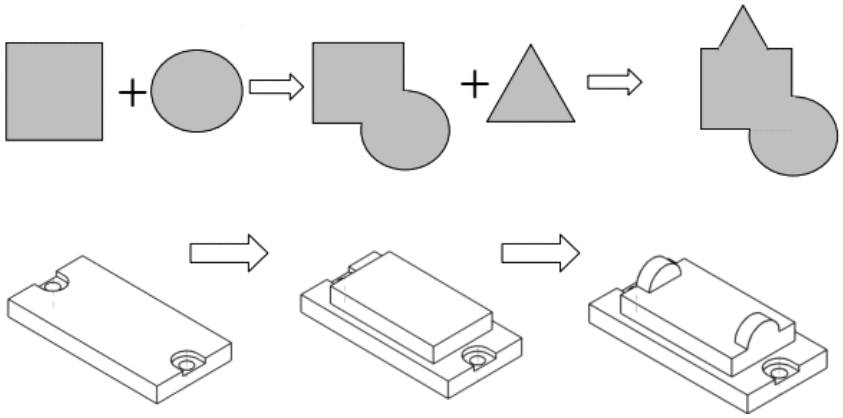


Рисунок 6.19 – Объединение элементов

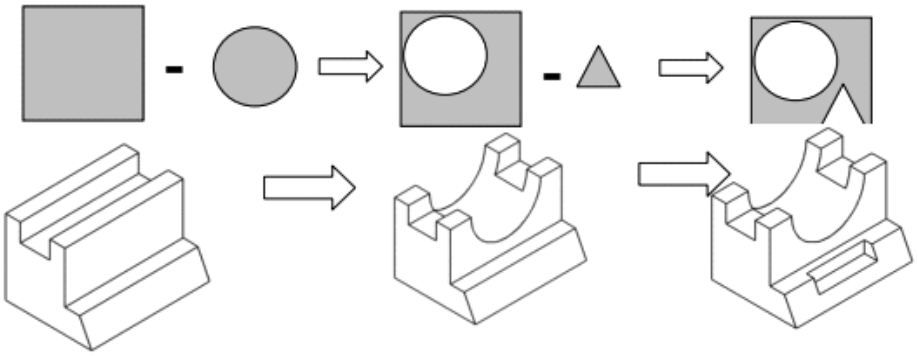


Рисунок 6.20 – Вычитание элементов

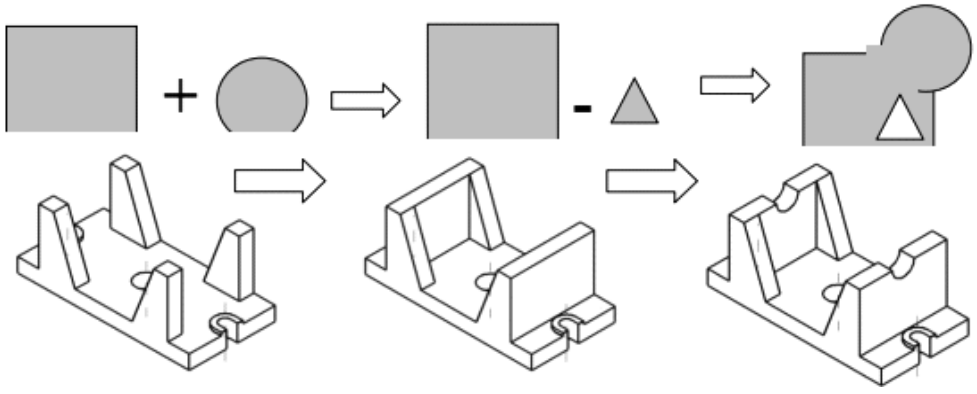
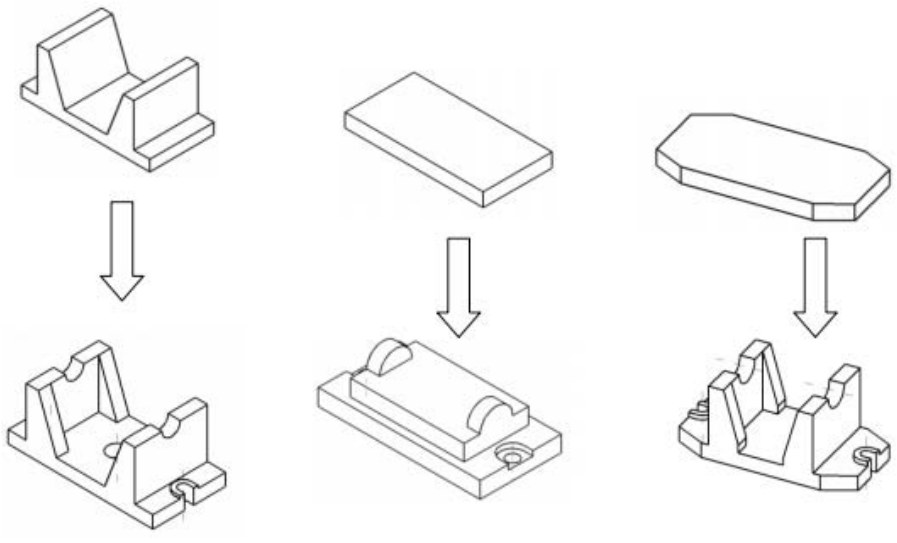


Рисунок 6.21 – Принцип построения деталей

Первоначальным элементом всех деталей является основание. На рис. 6.22,a,b,c показаны варианты оснований и деталей, образованных от них.



a *b*

Рисунок 6.22 – Детали и их основания

Построение трехмерного основания, а также других элементов детали в программе производят путем вытягивания двухмерного эскиза различными способами. На рис. 6.23 приведены варианты простой трансформации исходного двухмерного эскиза в трехмерную модель: а — методом «Бобышка вытянуть»; б— методом «Бобышка повернуть»

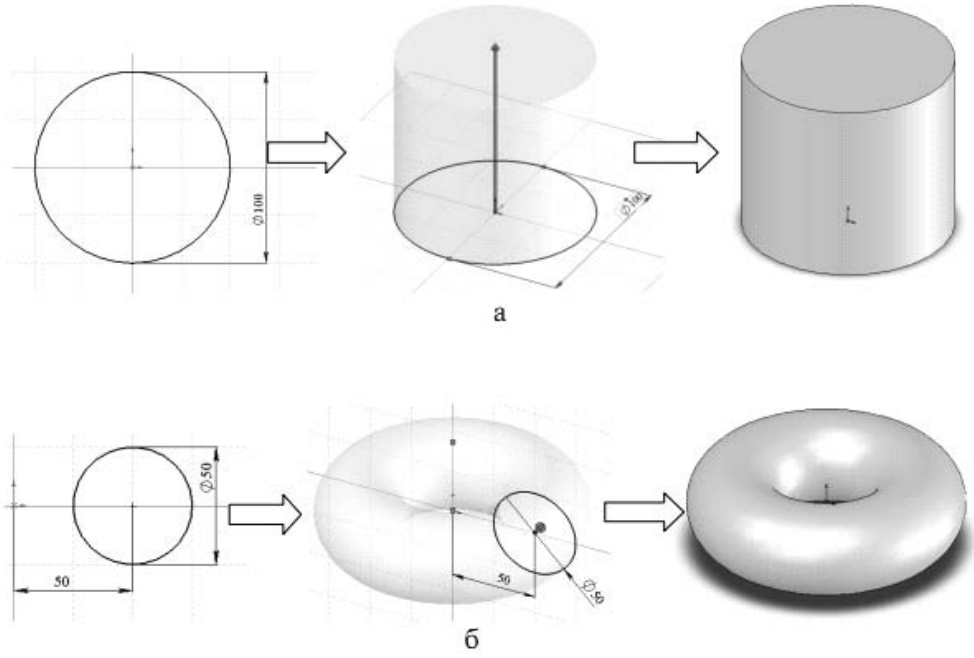


Рисунок 6.23 – Получение оснований

Существуют более сложные способы получения основания, например, путем вытягивания двухмерного эскиза вдоль кривой (рис. 6.24,а) или получение модели вытягиванием ее через различные контрольные сечения (рис. 6.24,б).

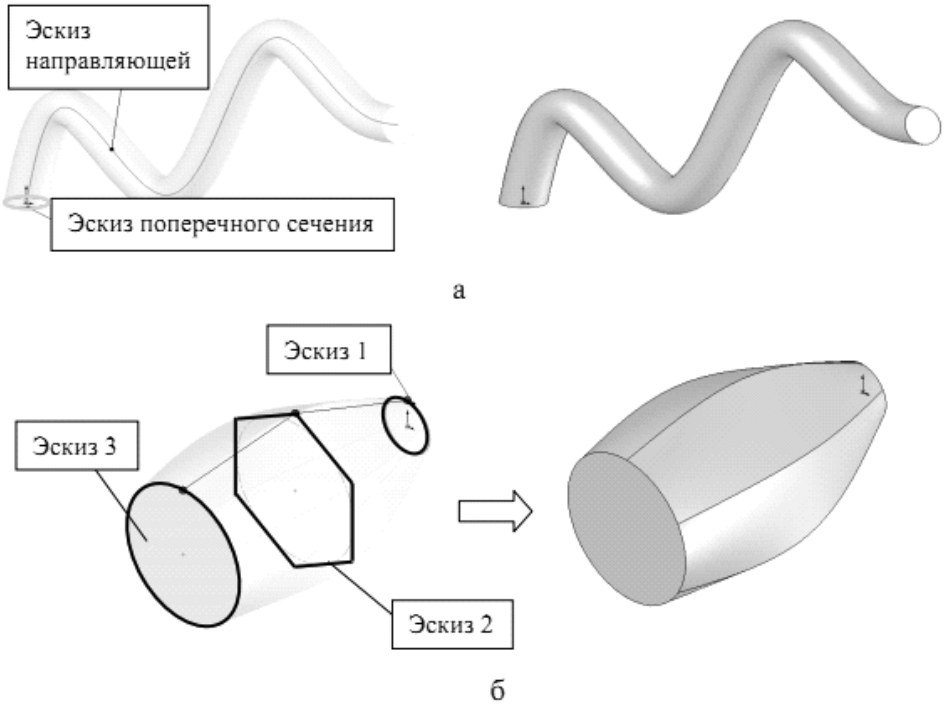


Рисунок 6.24

*6.3.2. **Методы проектирования сборки***

SolidWorks предлагает конструктору довольно гибкие возможности создания узлов и сборок. Система поддерживает как создание сборки способом «снизу вверх», т.е. на основе уже имеющихся деталей, число которых может доходить до сотен и тысяч, так и проектирование «сверху вниз».

Проектирование сборки начинается с задания взаимного расположения деталей друг относительно друга, причем обеспечивается предварительный просмотр накладываемой пространственной связи. Для цилиндрических поверхностей могут быть заданы связи концентричности, для плоскостей – их совпадение, параллельность, перпендикулярность или угол взаимного расположения.

Работая со сборкой, можно по мере необходимости создавать новые детали, определяя их размеры и расположение в пространстве относительно других элементов сборки. Наложенные связи позволяют автоматически перестраивать всю сборку при изменении параметров любой из деталей, входящих в узел. Каждая деталь обладает материальными свойствами, поэтому существует возможность контроля собираемости сборки. Для проектирования изделий, получаемых с помощью сварки, система позволяет выполнить объединение нескольких свариваемых деталей в одну (рис. 6.25).

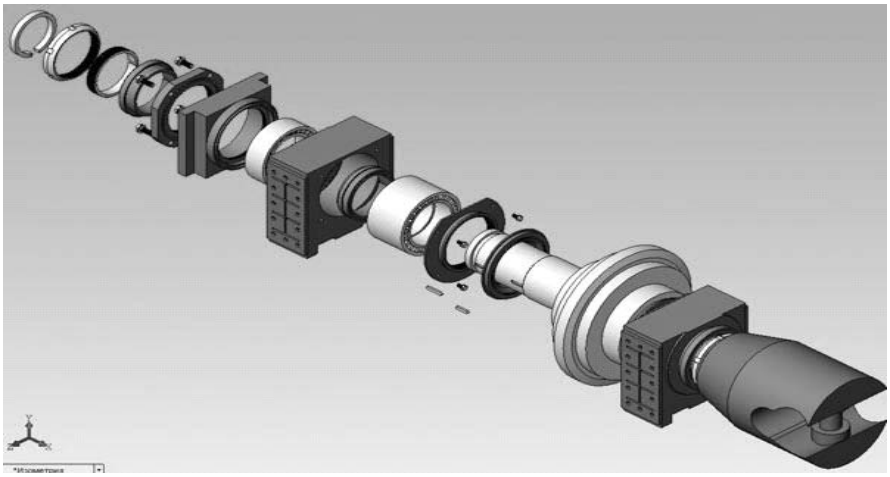


Рисунок 6.25 – Пример создания сборки с помощью элемента «концентричность»

Выбор метода проектирования элементов модели может быть различным в зависимости от геометрии проектируемого элемента и возможных методов его получения, а также последовательности действий при проектировании. На примере следующих нескольких способов показаны возможные варианты получения детали типа трубы.

Способ 1. Вытягивание цилиндра и вырез отверстия. На рис. 6.26 приведен процесс построения полого цилиндра.

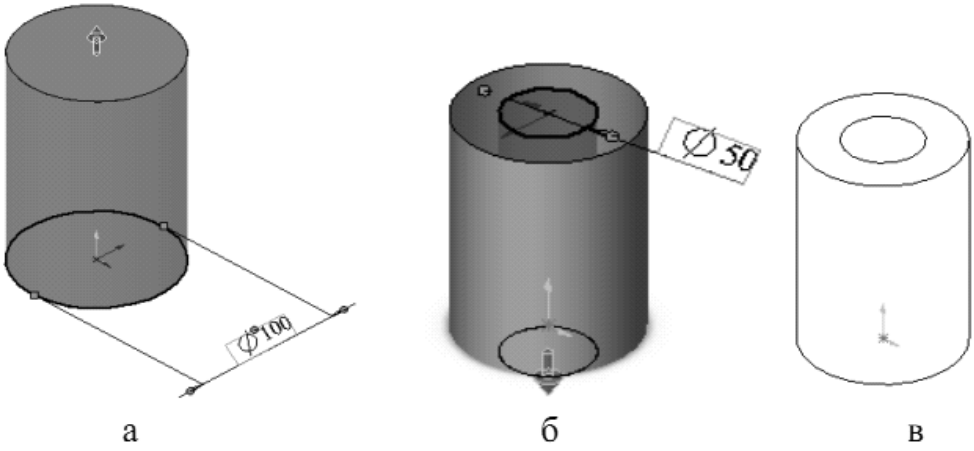


Рисунок 6.26 – Построение полого цилиндра: а – вытягивание окружности; б – создание выреза; в – результат

Способ 2. Вытягивание поперечного сечения трубки. На рис. 6.27 приведен процесс построения полого цилиндра указанным способом.

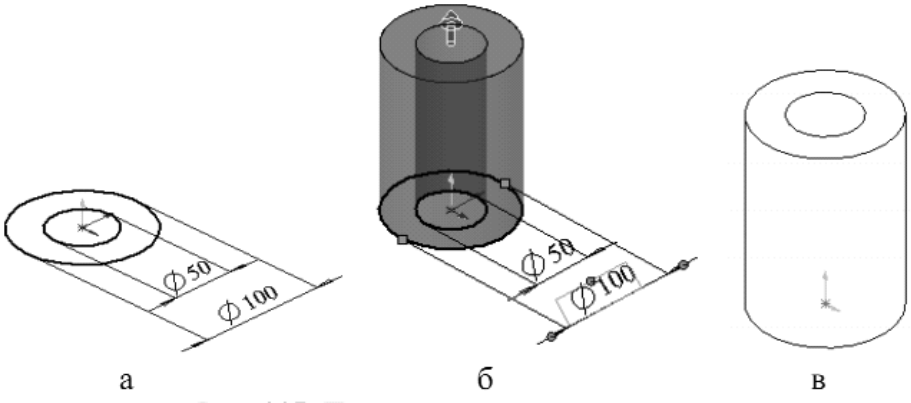


Рисунок 6.27 – Построение полого цилиндра: а – создание эскиза сечения; б – вытягивание созданного эскиза; в – результат

Способ 3. Создание цилиндра вращением и вырез отверстия вытягиванием. На рис. 6.28 приведен процесс построения полого цилиндра указанным способом.

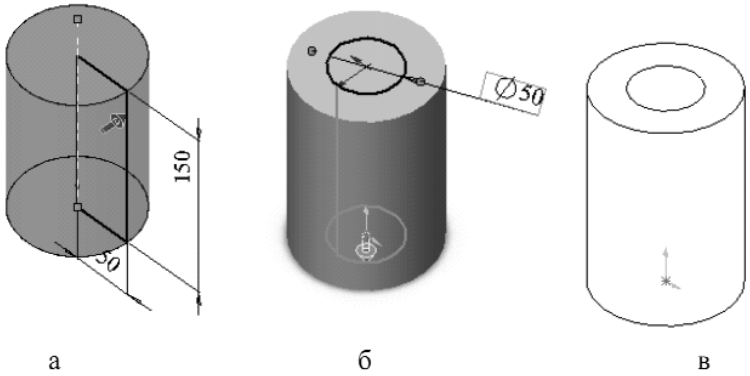


Рисунок 6.28 – Построение полого цилиндра: а – создание тела вращения – цилиндра; б – вырез внутренней полости; в — результат

Способ 4. Создание тела вращения и выреза вращения. На рис.6.29 приведен процесс построения полого цилиндра указанным способом.

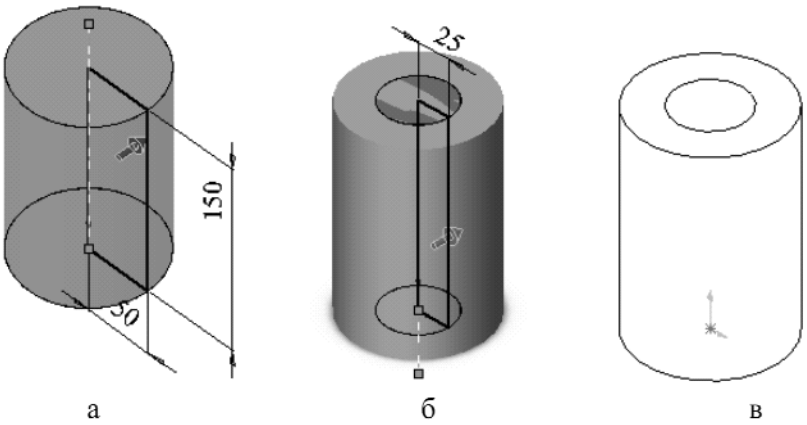


Рисунок 6.29 – Построение полого цилиндра: а – создание тела вращения – цилиндра; б – создание выреза вращения; в – результат

Способ 5. Создание тела вращения. На рис. 6.30 приведен процесс построения полого цилиндра указанным способом.

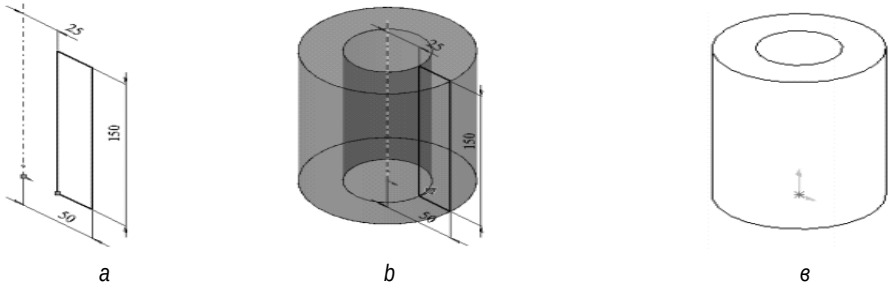


Рисунок 6.30 –Построение полого цилиндра: *а* – создание сечения вращения; *б* – создание тела вращения; *в* – результат

Обычно после первичного анализа и выбора плоскости проектирования основания происходит более детальная проработка модели с уточнением особенностей формы. Модель, состоящая из нескольких элементов, может быть построена в различном порядке их следования, а также различными методами. Следующий пример наглядно указывает на то, что порядок формирования независимых элементов детали может быть различным. На следующих примерах наглядно показано, что независимые по очередности построения элементы детали позволяют производить их формирование в различном порядке следования. Причем плоскость основания, количество построений, эскиз основания и конечный результат у деталей совпадают, что говорит о многовариантности выполнения проектирования. На приведенных на рис. 6.31 вариантах построения одной и той же детали видна многовариантность проектирования.

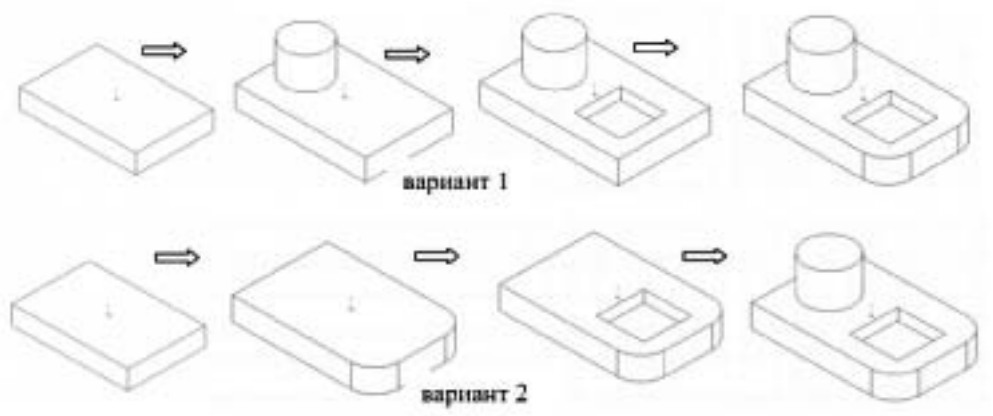


Рисунок 6.31 – Варианты построения модели детали

В следующих примерах на рис. 6.32 – 6.35 приведены варианты построения одной и той же модели разными способами, где видно, что можно выбирать для основания как разные плоскости, так и различные эскизы на одной и той же плоскости.

Способ 1 (рис. 6.32). Поэтапное создание детали путем добавления простых элементов. Изначальный выбор плоскости для построения основания – «Сверху».

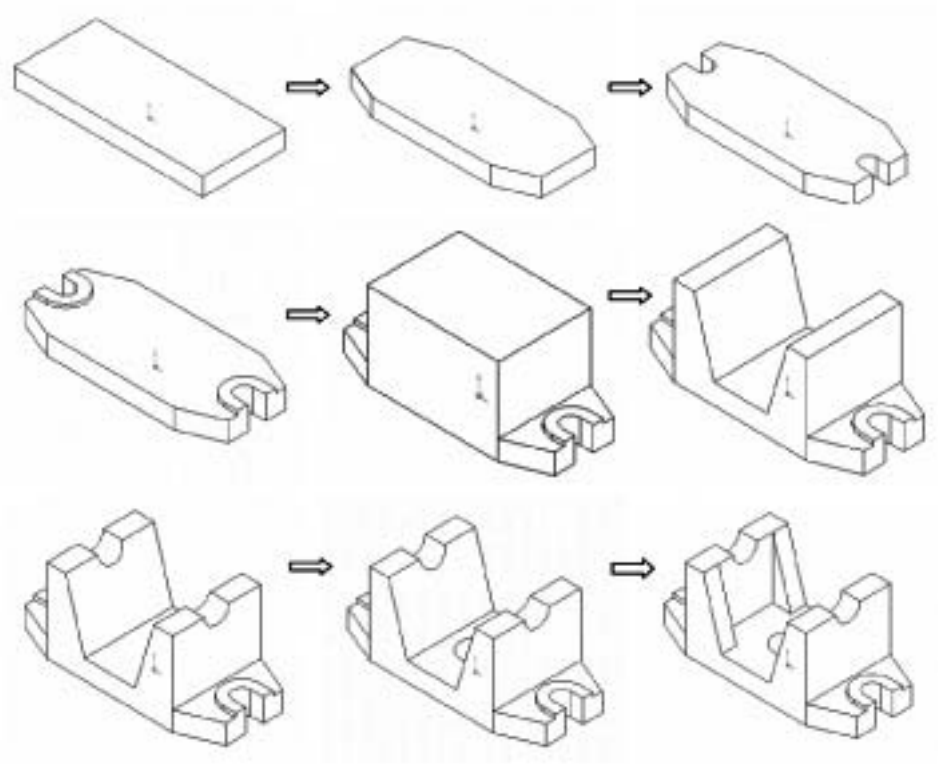


Рисунок 6.32 – Построение модели детали за 9 шагов

Способ 2 (рис. 6.33). Создание детали путем добавления более сложных элементов). Изначальный выбор плоскости для построения основания – «Сверху».

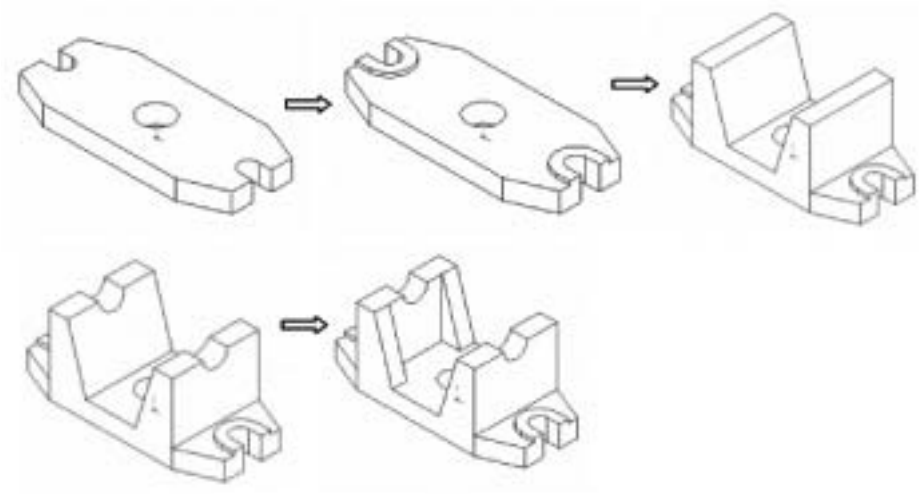


Рисунок 6.33 – Построение модели детали за 5 шагов

Способ 3 (рис. 6.34). Поэтапное создание детали путем создания сложного основания за 6 шагов. Изначальный выбор плоскости для построения основания – «Спереди».

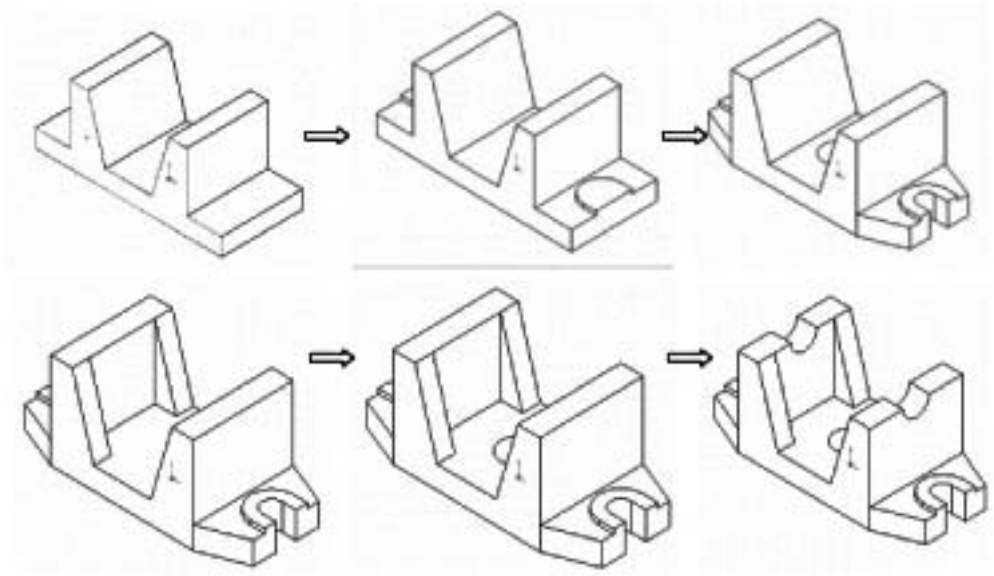


Рисунок 6.34 – Построение модели детали за 6 шагов

Способ 4 (рис. 6.35). Поэтапное создание детали путем добавления простых элементов. Изначальный выбор плоскости для построения основания – «Спереди».

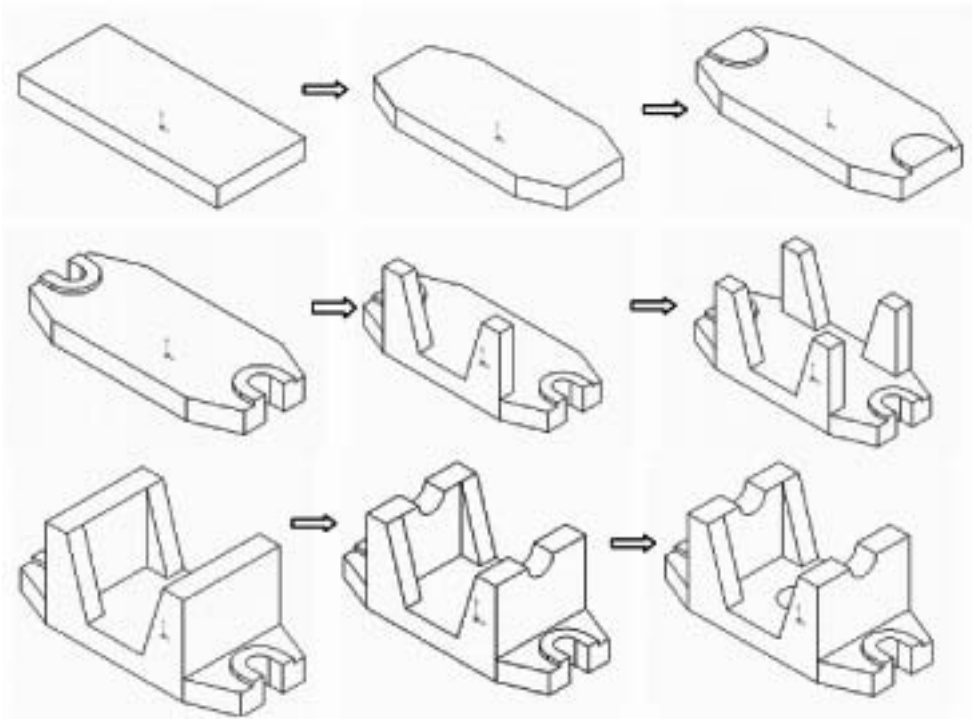


Рисунок 6.35 – Построение модели детали за 9 шагов

Из сравнения четырех вариантов построения детали видно, что существует более простой и понятный способ построения детали – поэлементно (варианты 1 и 4). Варианты 2 и 3 являются более сложными, но состоят из меньшего числа построений.

Выбор начальной плоскости проектирования и числа построений детали – индивидуальный выбор пользователя, однако следует помнить о том, что при любом проектировании должны присутствовать элементы оптимизации построения. Эффективно спроектированная деталь позволяет уменьшить число построений в Дереве конструирования и быть легкой в понимании и редактировании.

6.4 **Конструирование объектов с использованием пакета** AutoCAD

Системы графического моделирования — пакеты, связанные с построением и разработкой моделей графических образов.

AutoCAD – это двух- и трёхмерная система автоматизированного проектирования и черчения (САПР), разработанная компанией Autodesk. Графическая система *AutoCAD* (Автоматизированное компьютерное черчение и проектирование) – практически мировой стандарт в области систем автоматизированного проектирования для персональных компьютеров. Стандарты *AutoCAD* поддерживают огромное число независимых разработчиков, создавших более 5000 специализированных приложений для *AutoCAD* во всех прикладных областях. *AutoCAD* применяется практически во всех отраслях промышленности и строительства, так как в последнее время заказчики стали предъявлять требования к проектировщикам по выпуску документации в электронном виде. *AutoCAD* относится к классу программ CAD (Computer Aided Design), которые

предназначены, в первую очередь, для разработки конструкторской документации: чертежей, моделей объектов, схем и т. д.

Программа позволяет строить 2D и 3D чертежи любых назначения и сложности с максимальной точностью. Скорость и легкость, с которыми создаются трехмерные модели проектируемых изделий, широкие возможности их преобразования и редактирования, различные способы получения плоских изображений этих изделий (видов, разрезов, сечений), ассоциативно связанных с моделями, – все это обеспечивает огромную экономию времени по сравнению с «ручным» черчением.

Современный пакет *AutoCAD* позволяет работать одновременно с несколькими чертежами, имеет мощные средства визуализации создаваемых трехмерных объектов и расширенные возможности адаптации системы к требованиям пользователя, обеспечивает связь графических объектов с внешними базами данных, позволяет просматривать и копировать компоненты чертежа без открытия его файла, редактировать внешние ссылки и блоки, находящиеся во внешних файлах.

В недавно вышедшей версии *AutoCAD 2015* появились новые инструменты: усовершенствованный интерфейс, улучшенные онлайн-карты, синхронизация чертежей, пакет инструментов для черчения, предварительный просмотр команд, галереи ленты.

AutoCAD и специализированные приложения на его основе нашли широкое применение в машиностроении, строительстве, архитектуре и других отраслях промышленности.

Специализированные приложения на основе *AutoCAD:*

- AutoCAD Architecture — версия, ориентированная на архитекторов и содержащая специальные дополнительные инструменты для архитектурного проектирования и черчения, а также средства выпуска строительной документации.
- AutoCAD Electrical разработан для проектировщиков электрических систем управления и отличается высоким уровнем автоматизации стандартных задач и наличием обширных библиотек условных обозначений.
- AutoCAD Civil 3D — решение для проектирования объектов инфраструктуры, предназначенное для землеустроителей, проектировщиков генплана и проектировщиков линейных сооружений. Помимо основных возможностей, AutoCAD Civil 3D может выполнять такие виды работ, как геопространственный анализ для выбора подходящей стройплощадки, анализ ливневых стоков для обеспечения соблюдения экологических норм, составление сметы и динамический расчет объёмов земляных работ.
- AutoCAD MEP ориентирован на проектирование инженерных систем объектов гражданского строительства: систем сантехники и канализации, отопления и вентиляции, электрики и пожарной безопасности. Реализовано построение трехмерной параметрической модели, получение чертежей и спецификаций на её основе.
- AutoCAD Map 3D создан для специалистов, выполняющих проекты в сфере транспортного строительства, энергоснабжения, земле- и водопользования и позволяет создавать, обрабатывать и анализировать проектную и ГИС-информацию.
- AutoCAD Raster Design — программа векторизации изображений, поддерживающая оптическое распознавание символов (OCR).
- AutoCAD Structural Detailing — средство для проектирования и расчёта стальных и железобетонных конструкций, поддерживающее технологию информационного моделирования зданий. Базовыми объектами являются балки, колонны, пластины и арматурные стержни и др.
- AutoCAD Ecscad позволяет инженерам-электрикам создавать схемы электротехнического оборудования с помощью сценариев и библиотек условных обозначений.
- AutoCAD Mechanical предназначен для проектирования в машиностроении и отличается наличием библиотек стандартных компонентов (более 700 тысяч элементов), генераторов

компонентов и расчётных модулей, средств автоматизации задач проектирования и составления документации, возможностью совместной работы.

- AutoCAD P&ID — это программа для создания и редактирования схем трубопроводов и КИП, а также для управления ими.
- AutoCAD Plant 3D — инструмент для проектирования технологических объектов. В AutoCAD Plant 3D интегрирован AutoCAD P&ID.

*6.4.1. **Интерфейс программы** AutoCAD*

Интерфейс программы *AutoCAD* представляет собой: панель инструментов, панель быстрого доступа, видовой куб, панель навигации, ленту, навигацию по листам, командную строку, строку состояния и пр.

Окно чертежа занимает наибольшую часть экрана. Именно в нем происходит построение чертежей. В окне программы может быть открыто несколько окон чертежа (рис. 6.36).

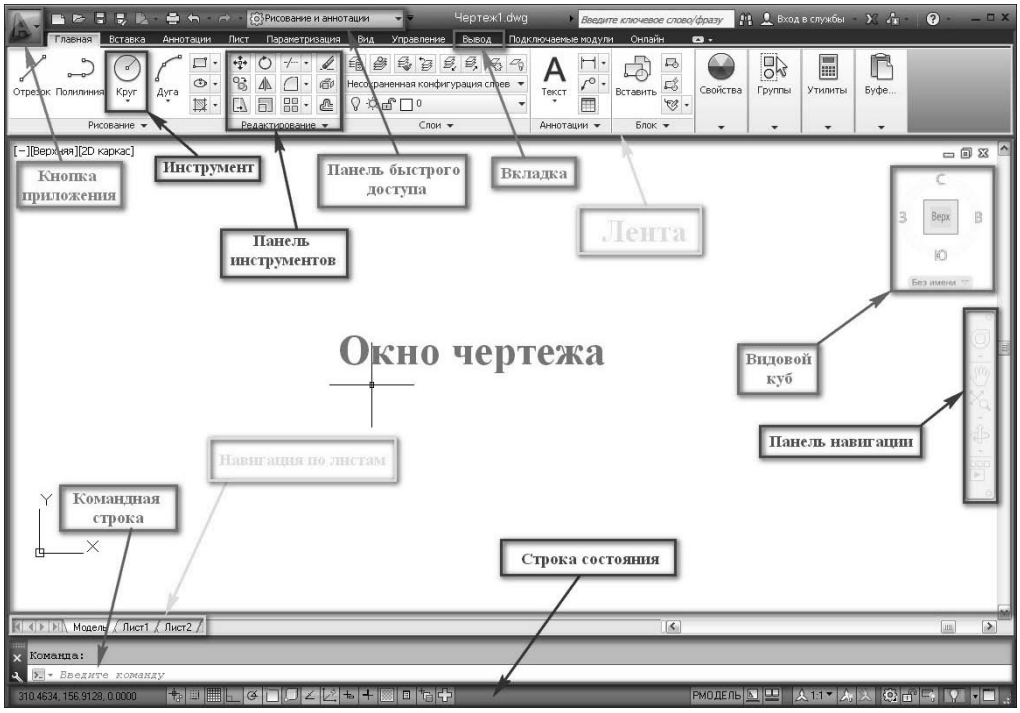


Рисунок 6.36 — Несколько окон чертежа (чертежей)

В левом верхнем углу расположена кнопка приложения. Если щелкнуть левой кнопкой мыши по этой кнопке, появится окно, позволяющее создавать, открывать и сохранять чертежи, а также осуществлять их публикацию и печать.

Кнопка параметры открывает окно «Настройка» (рис 6.37).

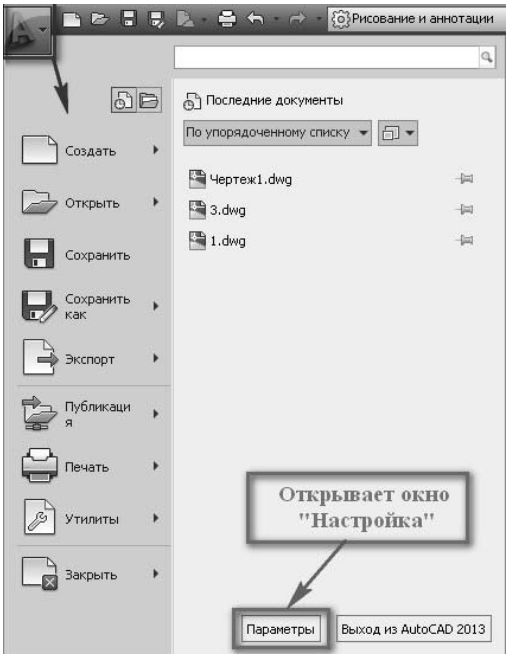


Рисунок 6.37 — Окно «Настройка»

По умолчанию лента отображается при открытии файла и представляет собой компактную палитру со всеми инструментами (рис. 6.38), необходимыми для создания или изменения чертежа.

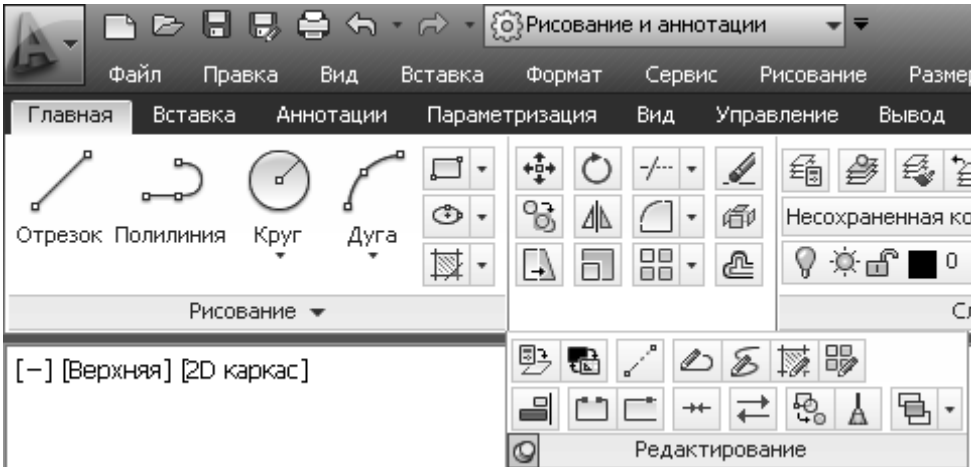


Рисунок 6.38 — Компактная палитра со всеми инструментами

Лента состоит из ряда панелей, которые систематизированы в виде вкладок, помеченных названием задачи. Панели ленты содержат многие из тех же инструментов и элементов управления, которые доступны на панелях инструментов и в диалоговых окнах.

6.4.2. ***Постановка задачи***

По предложенной схеме здания (рис. 6.39) выполнить:

- чертеж плана здания в масштабе 1:100;
- проставить размеры.

Исходные данные

Основные элементы здания:

Название элемента	Материал	Примечание
Наружные стены	кирпичные	Толщина стены - 640 мм, привязка[1] - 200/440
Внутренние стены	кирпичные	Толщина стены - 380 мм, привязка - 190/190
Перегородки	кирпичные	Толщина - 120 мм
Наружные лестницы	железобетонные ступени по железобетонным косоурам	Ширина проступи -300 мм, высота подступенка -150мм
Внутренние лестницы	деревянные	Индивидуальный проект

Оконные проемы с четвертями, оконные блоки с двойным остеклением. Ширина оконных и дверных проемов выбирается из таблицы, приведенной выше, согласно предложенной схеме здания.

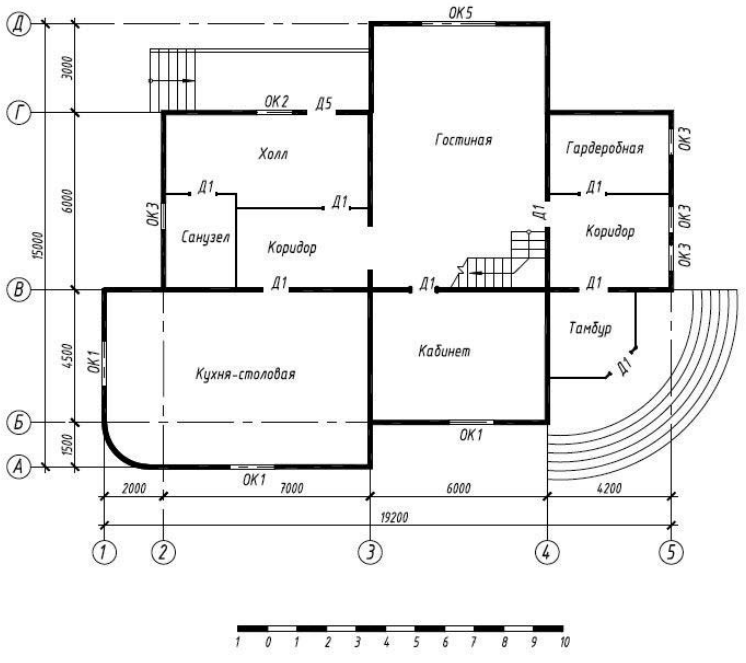


Рисунок 6.39 — Схема здания

Порядок выполнения работы

1. Создание слоёв

Создать слои, согласно рис. 6.40. Каждому слою присвоить имя, цвет, тип линии, вес линии.

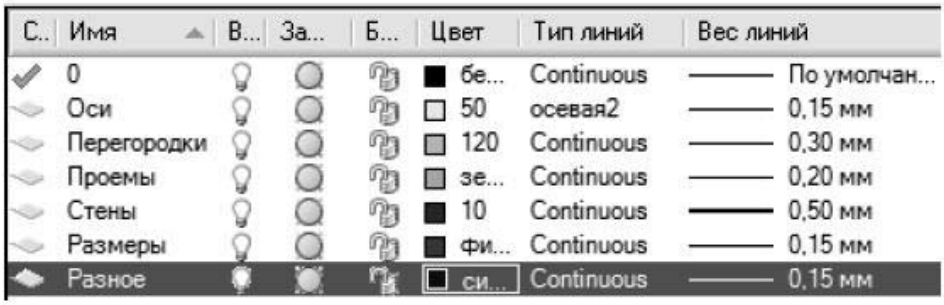


Рисунок 6.40 — Создание слоев

2. Вычерчивание координационных осей

Установить текущий слой *«Оси»*. Вычертить оси (рис. 6.41), используя команду «Отрезок».

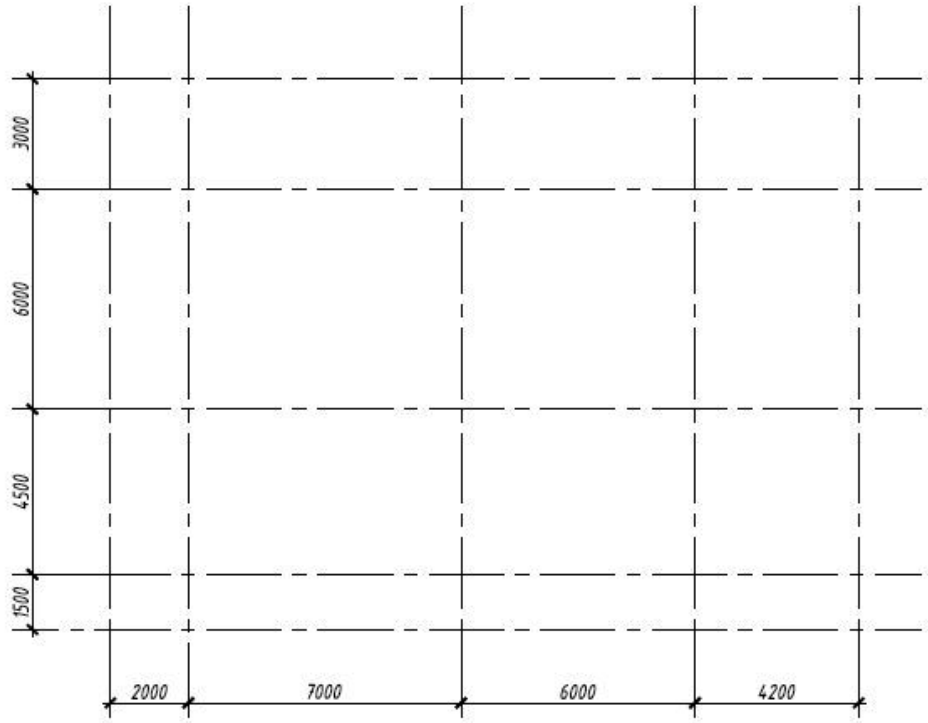


Рисунок 6.41- Вычерчивание оси

3. Вычерчивание внешних стен

Установить текущий слой «Стены». Вычертить командой. «Полилиния» (нулевой ширины) вспомогательный контур стен по координационным осям.

Наклонные участки контура стен строятся с использованием команды «Фаска», скругленные – команды «Сопряжение».

Для задания толщины наружных (рис. 6.42) стен необходимо использовать команду «Подобие».

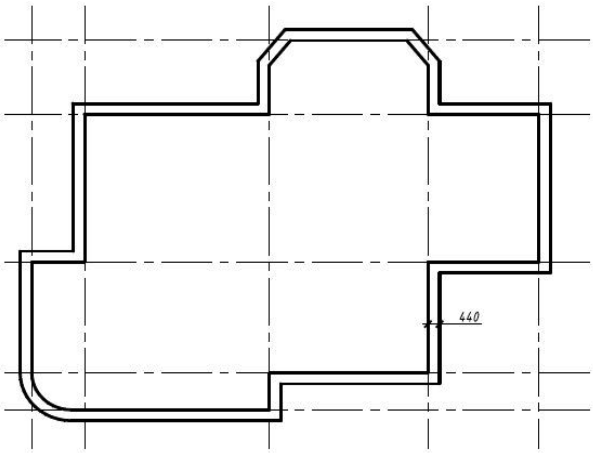


Рисунок 6.42 — Задания толщины наружных стен

Вычерчивание внутренних стен и перегородок

Установить текущий слой «Стены». Вычертить командой «Мультилиния». Для вычерчивания перегородок установить текущий слой «Перегородки». Вычертить командой «Мультилиния». Для редактирования пересечения стен и перегородок необходимо использовать команды: «Обрезать» и «Редактирование мультилиний» (рис. 6.43).

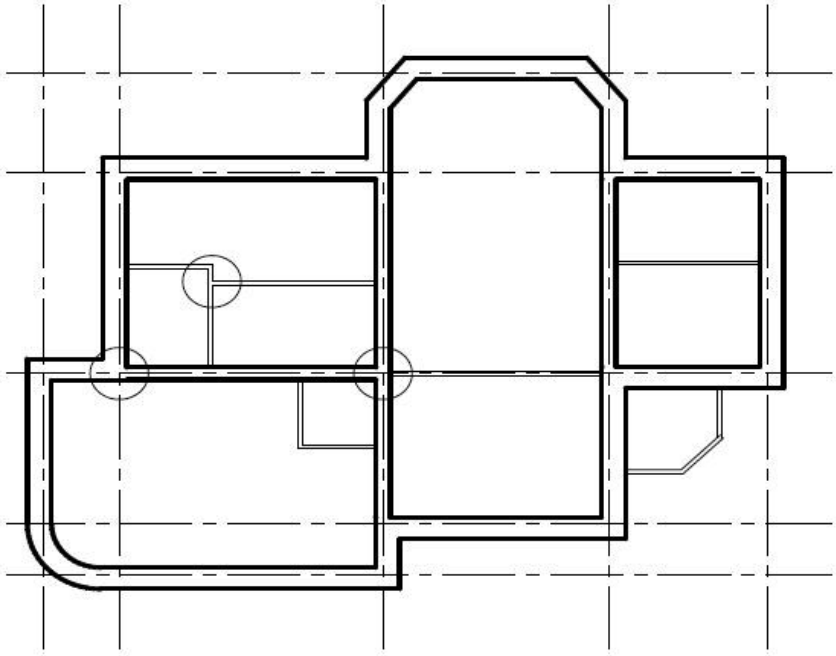


Рисунок 6.43 — Команды: «Обрезать» и «Редактирование мультилиний»

5/ Вычерчивание оконных проемов

Вычертить контур оконного проёма на свободном поле чертежа с использованием команды «Отрезок» в следующих слоях: боковые линии – слой «Стены», горизонтальные – «Проемы» (рис. 6.44).

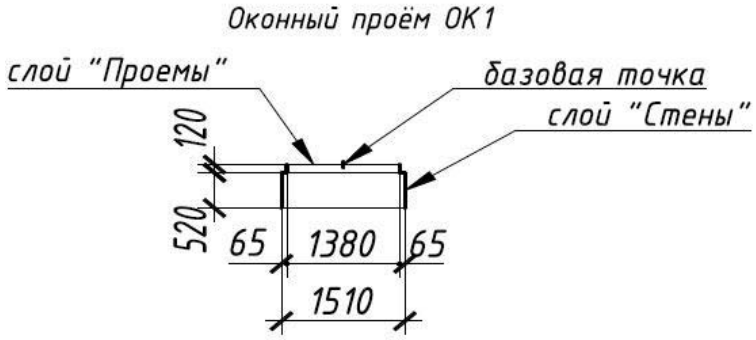


Рисунок 6.44

Отредактировать места вставки оконных проемов – обрезать участок стены, используя команду «Разорвать»(рис. 6.45).

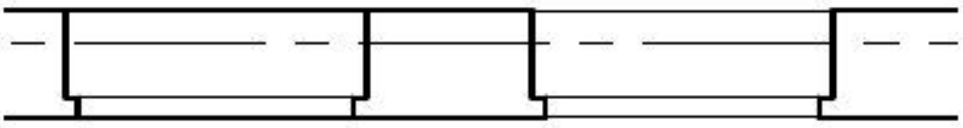


Рисунок 6.45

6. Вычерчивание дверных проемов

Вычертить контур дверного проема при помощи команды «Отрезок» (рис. 6.46) в следующих слоях: боковые линии – слой *«Стены» или «Перегородки»* в зависимости от того, где расположена дверь, створки – *«Проемы».* Затем отредактировать места вставки дверных проемов с использованием команды «Обрезать».

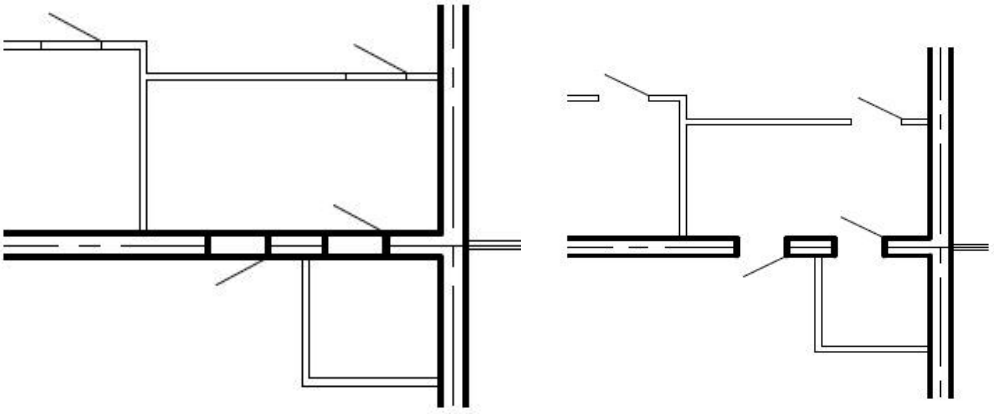


Рисунок 6.46

7. Вычерчивание внутренних и наружных лестниц (рис. 6.47)

Рассчитать лестничные марши. Установить текущий слой «*Разное».* Вычертить лестничные марши с использованием команд: «Отрезок», «Дуга», «Подобие», «Массив».

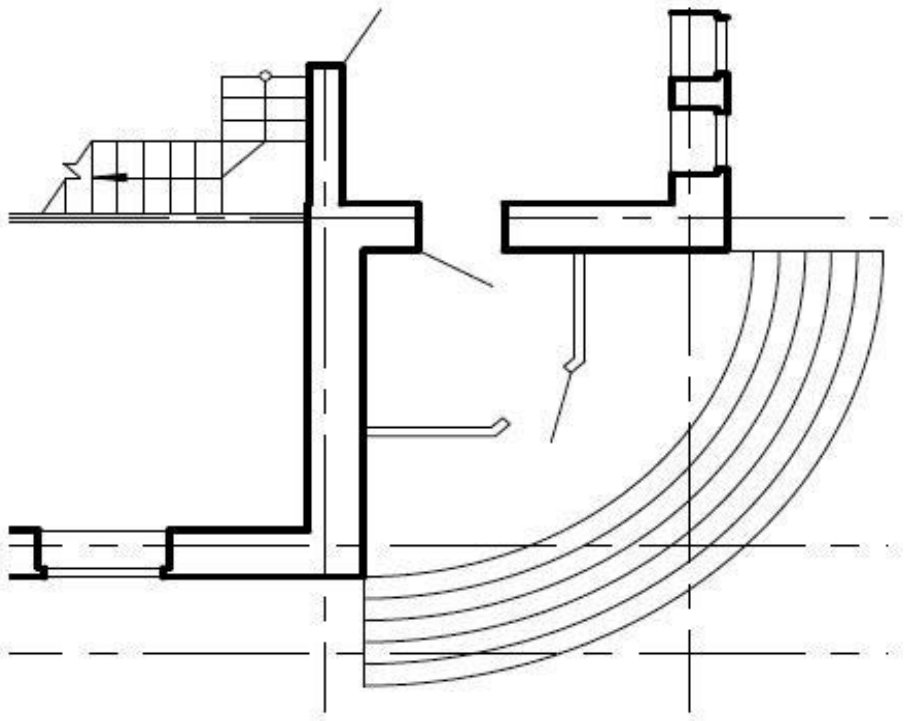


Рисунок 6.47 — Вычерчивание внутренних и наружных лестниц

8. Расстановка сантехнического оборудования(рис. 6.48)

Установить текущий слой «Разное». Использовать команды «Центр управления» (библиотека AutoCAD) и «Вставка блока», расположить сантехническое оборудование.

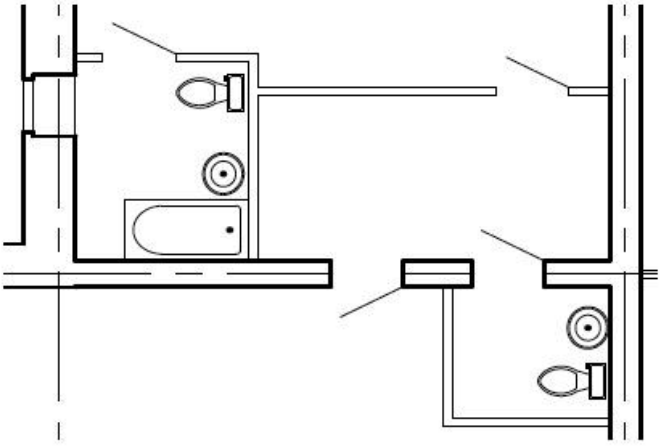


Рисунок 6.48 — Расстановка сантехнического оборудования

9. Простановка размеров (рис. 6.49)

Сделать текущим слой «Размеры». Создать новый размерный стиль – команда «Размерный стиль». Размеры проставить с использованием команд: «Линейный», «Продолжить», «Площадь». Промаркировать оси. Использовать команды «Круг», «Текст».

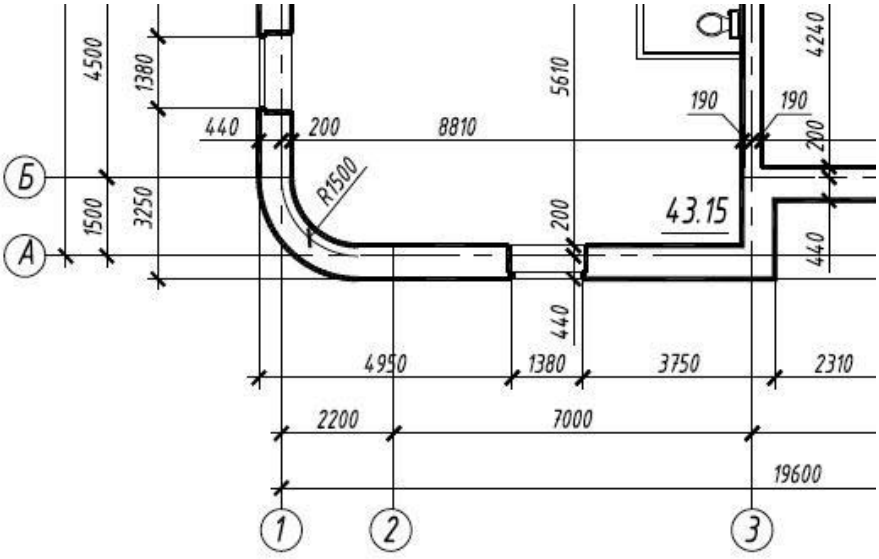


Рисунок 6.49 — Простановка размеров

Окончательное оформление чертежа (рис. 6.50)

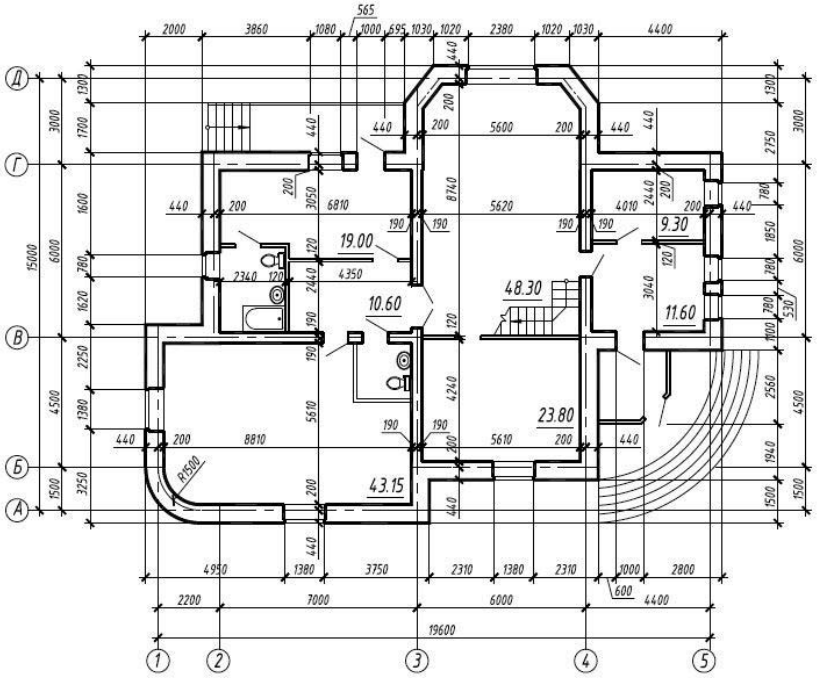


Рисунок 6.50 — Окончательное оформление чертежа

6.5. **Конструирование объектов с использованием пакета** GoogleSketchUp

При открытии *SketchUp* сначала мы видим окно приветствия (рис. 6.51). Где нужно выбрать шаблон рабочей области. Основные отличия этих шаблонов в единицах измерения (дюймы, метры и т.д.), и стартовых видах (сверху, снизу и т.д.).

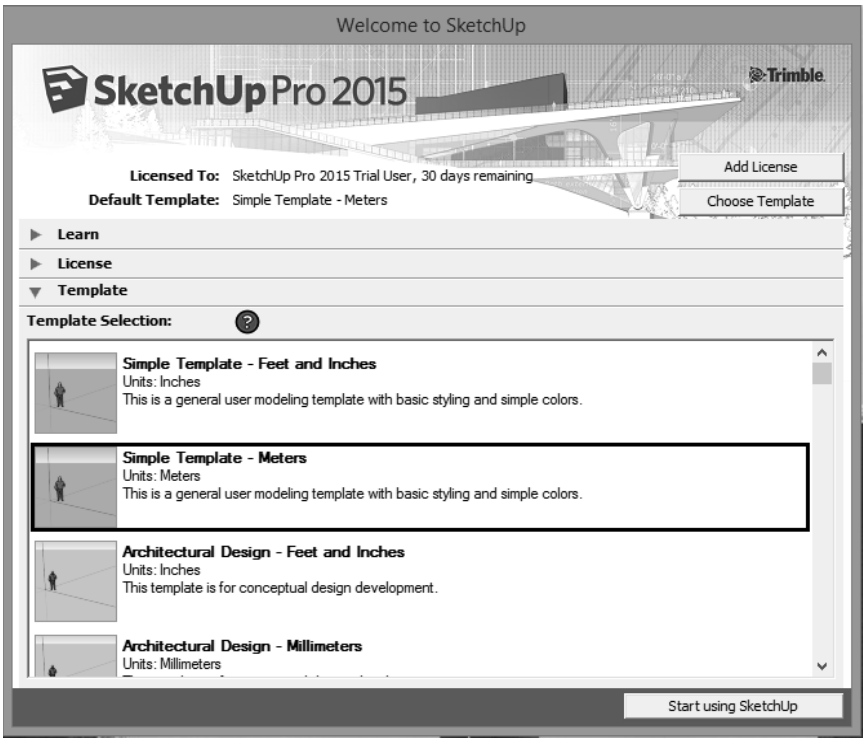


Рисунок 6.51 – Окно приветствия

Выбираем второй пункт SimpleTemplate – Meters. Нажимаем кнопку StartusingSketchUp. Попадаем в рабочую область (рис. 6.52).

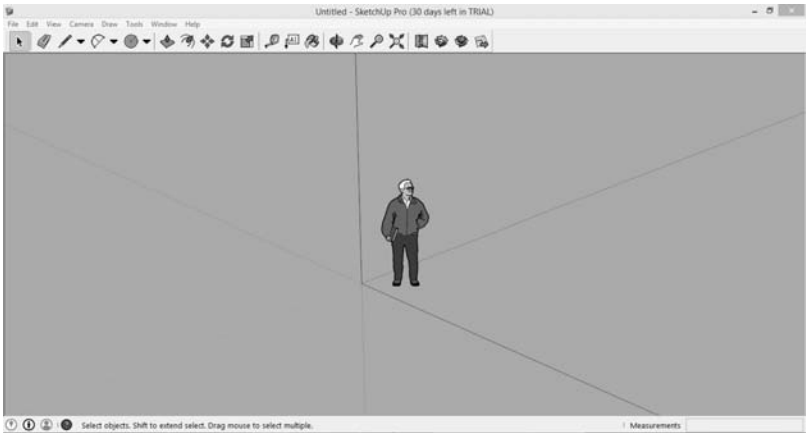


Рисунок 6.52 – Рабочая область

Все функции, которые нам понадобятся, находятся в верхней части окна. Внизу окна находится строка состояния. В ней мы можем наблюдать информацию о ходе выполнения каких-то команд. Для управления камерой используется мышка. Для поворота камеры используется зажатая средняя кнопка мыши (СКМ), для перемещения камеры – shift+СКМ, для приближения/отдаления камеры нужно прокрутить СКМ.

Удалим человека в центре координат (рис. 6.53). Для этого выделим его с помощью мыши и нажмем клавишу DEL.

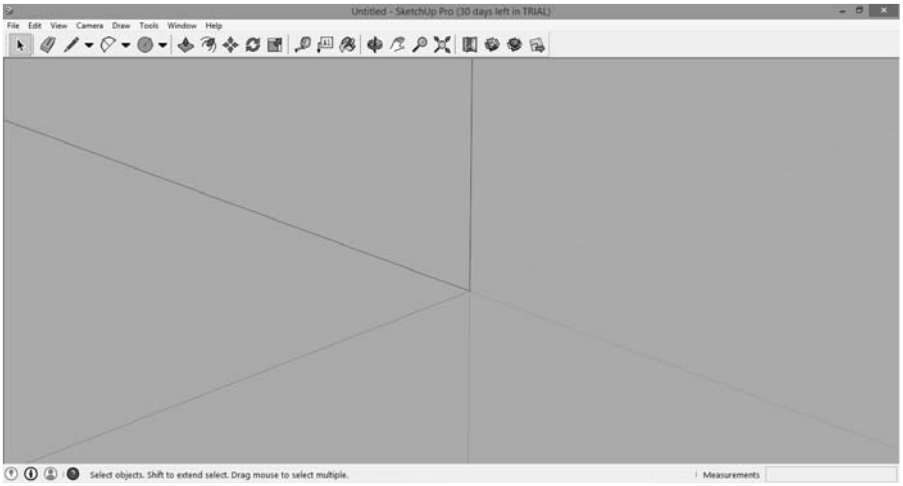


Рисунок 6.53 — Удаление человека в центре координат

Выберем инструмент «Rectangle»  (рис. 6.54) в выпадающем списке «Shapes»  и нарисуем его на рабочей области. Далее выбираем команду «Push/Pull»  , после чего кликаем по прямоугольнику и тянем наверх, у нас получается коробочка. Интересно попрактиковаться с данным инструментом.

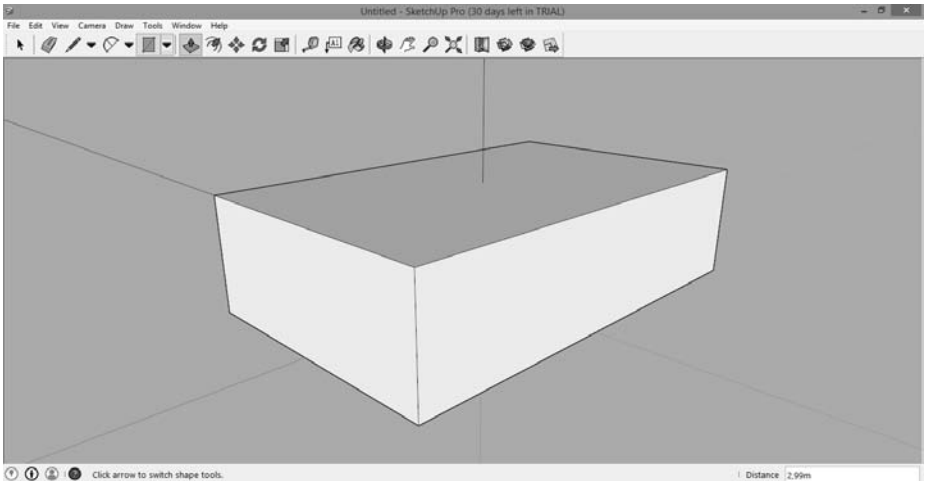


Рисунок 6.54 – Инструмент «Rectangle»

Теперь выберем инструмент «Circle»  (рис. 6.55) и нарисуем его на поверхности нашей коробки, далее выдавим его с помощью уже знакомой нам команды «Push/Pull»  .

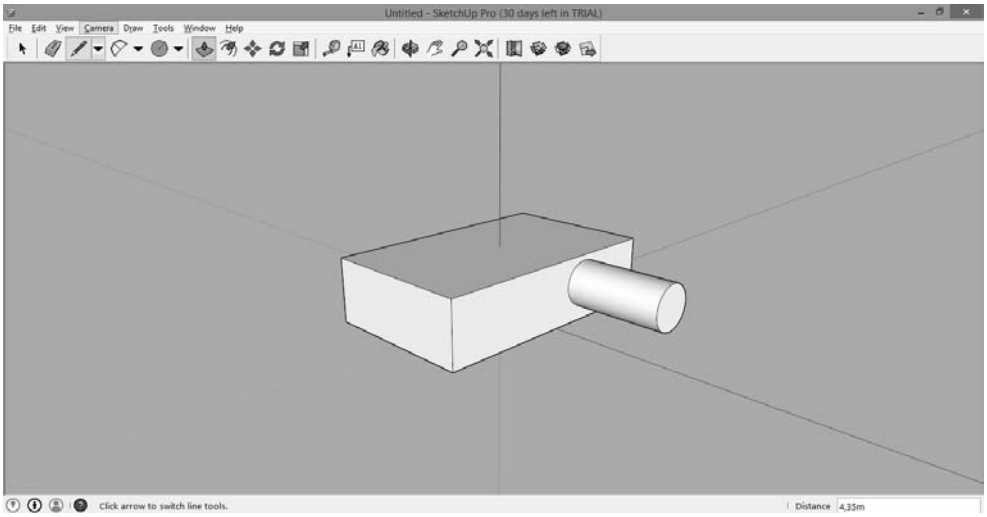


Рисунок 6.55 — Инструмент «Circle»

Поэкспериментируем с командами «Circle», «Rectangle», «Push/Pull». Результат использования команд показан на рис. 6.56. Несложно по аналогии самостоятельно разобраться с остальными командами SketchUp.

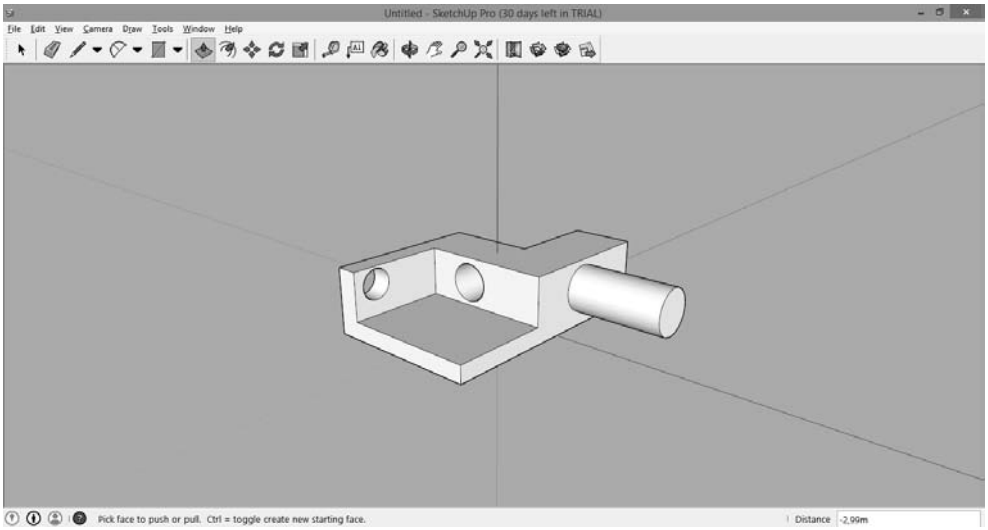


Рисунок 6.56

Создать модель дома, изображенную на картинке (рис. 6.57):

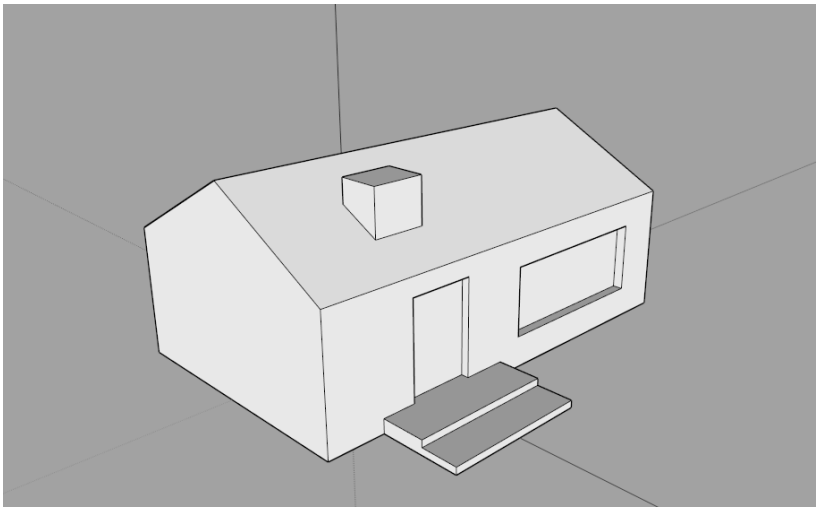


Рисунок 6.57 — Создание модели дома

Нарисуем прямоугольник  и с помощью инструмента «Push/Pull»  вытянем его в коробку. Получим примерно такое изображение (рис. 6.58).

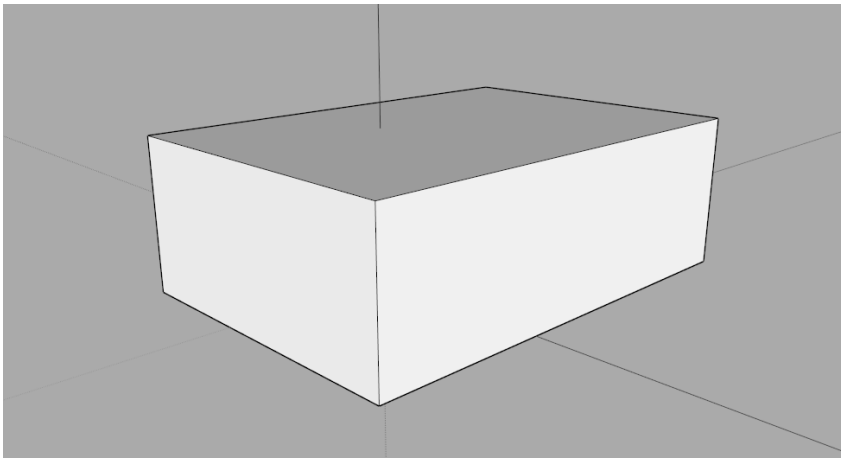


Рисунок 6.58 — Коробка

Разделим верхнюю грань с помощью линии  на две равные части. Для этого надо найти «midpoint» на грани (подсвечивается голубым цветом). Делим эту линию с помощью инструмента выделения  , после чего данную грань можно как угодно перемещать с помощью инструмента перемещения  . Получим примерно такую модель (рис. 6.59):

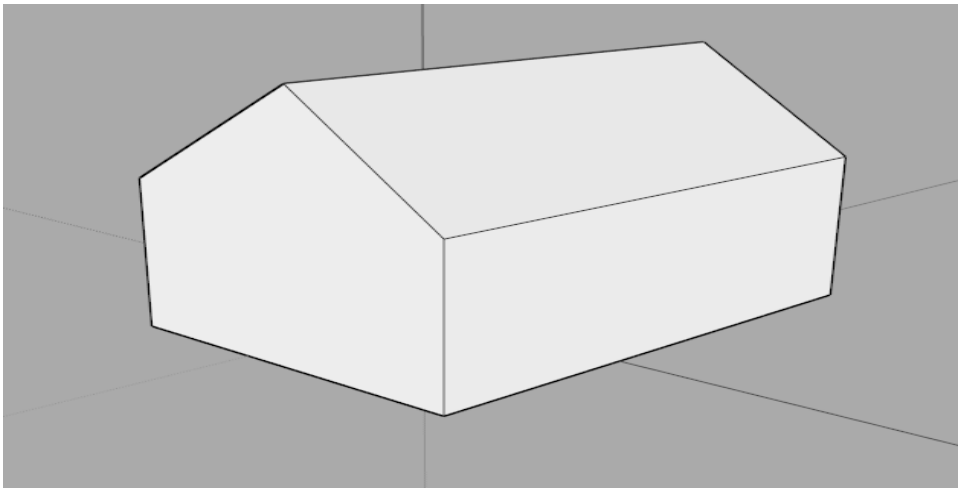


Рисунок 6.59

Далее с помощью линии нарисуем контур трубы (дымаря), для этого стоит обращать внимание на подсказки среды моделирования (при проведении линии вдоль каждой из осей она подсвечивается соответствующим цветом: синий, зелены, красный). После этого с помощью инструмента «Push/Pull» вытянем его до нужной нам формы. Получим примерно такое изображение (рис. 6.60):

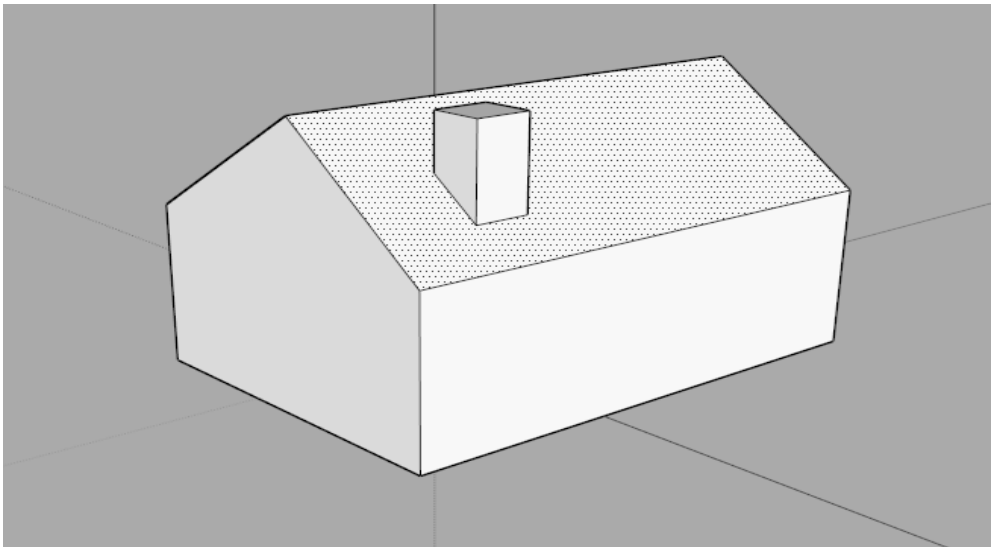


Рисунок 6.60 — Инструмент «Push/Pull»

Теперь добавим наши модели деталей. Нарисуем с помощью прямоугольника ступени, дверь и окно. Получим такое изображение (рис. 6.61):

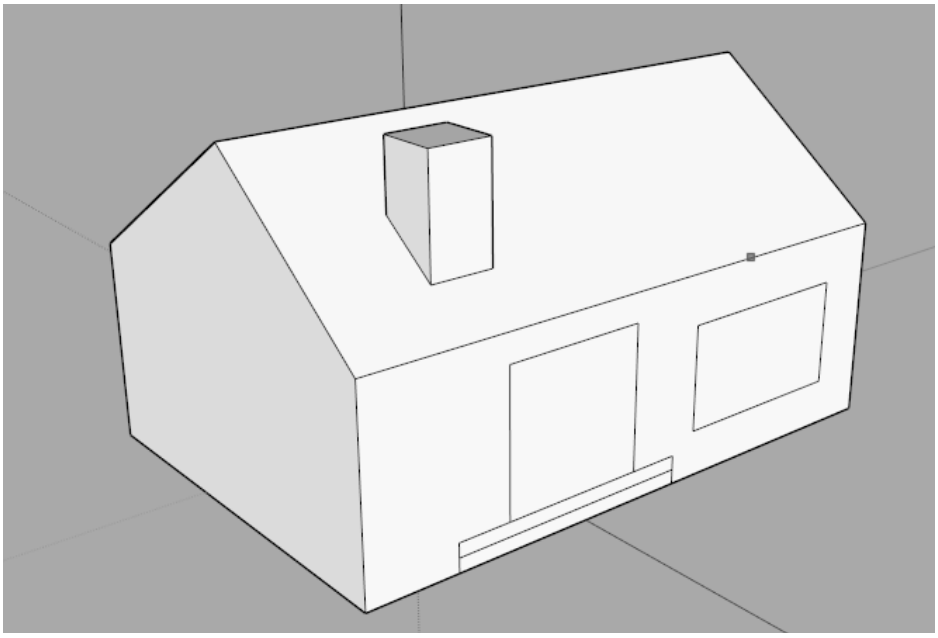


Рисунок 6.61

С помощью инструмента «Push/Pull» (вытянем/вдавим) соответствующие элементы дома. Конечный вариант домика получится такой (рис. 6.62):

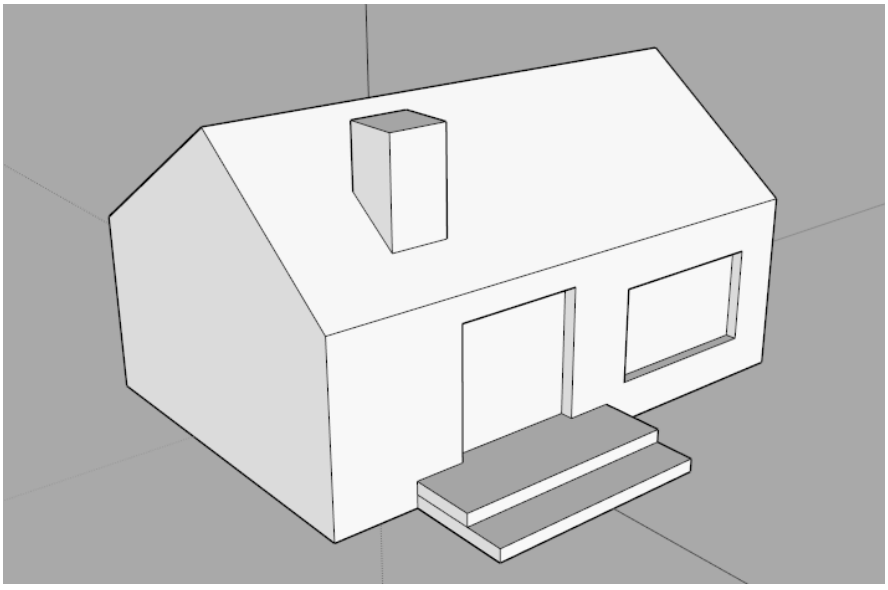


Рисунок 6.62 — Инструмент «Push/Pull»

6.6. **Конструирование объектов с использованием пакета Компас**

«Ко́мпас» — семейство систем автоматизированного проектирования с возможностями оформления проектной и конструкторской документации согласно стандартам серии ЕСКД и СПДС. Разрабатывается российской компанией «Аскон». Название линейки является акронимом от фразы «КОМПлекс Автоматизированных Систем», в торговых марках используется написание заглавными буквами. Первый выпуск «Компаса» (версия 1.0) состоялся в 1989 году. Первая версия под Windows — «Компас 5.0» — вышла в 1997 году.

Программы данного семейства автоматически генерируют ассоциативные виды трёхмерных моделей (в том числе разрезы, сечения, местные разрезы, местные виды, виды по стрелке, виды с разрывом). Все они ассоциированы с моделью: изменения в модели приводят к изменению изображения на чертеже. Стандартные виды автоматически строятся в проекционной связи. Данные в основной надписи чертежа (обозначение, наименование, масса) синхронизируются с данными из трёхмерной модели. Имеется возможность связи трёхмерных моделей и чертежей со спецификациями, то есть при «надлежащем» проектировании спецификация можетбыть получена автоматически; кроме того, изменения в чертеже или модели будут передаваться в спецификацию, и наоборот.

Создание нового чертежа в программе *Компас* начинается с задания лимитов и начальных настроек чертежа:

1. Выставляем лимиты чертежа Шаблоны\Констр. Чертеж А3 горизонтальный первый лист, соответствующий формату А3.

2. Сохраняем чертежа на жёсткий диск, присваиваем ему имя Валик.cdw (для того, чтобы в процессе работы производить сохранение чертежа одним щелчком мыши на кнопке Сохранить).

3. Создаем рамку и основную надпись чертежа (рис. 6.63). Поскольку в среду *Компас* занесены все стандарты, то выполним основную задачу по созданию рамки, осталось только вписать нужные данные в штамп.

Создаем новый слой Текст и заполняем штамп текстом.

Рамка и основная надпись готовы!

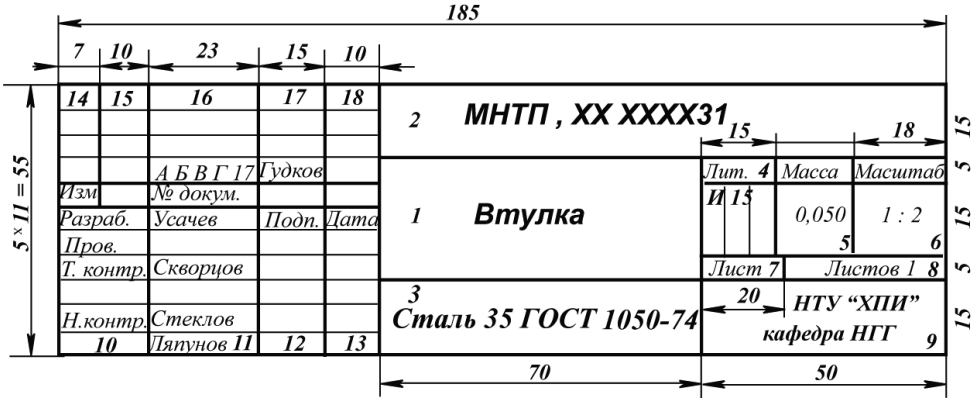


Рисунок 6.63 — Рамка и основная надпись чертежа

Начертить саму детали можно тремя способами: разбить очертания детали на примитивы (прямоугольники, квадраты, линии, круги, дуги и т. д.), начертить эти самые примитивы с нужными размерами и, в конечном счёте, составить деталь из этих элементов; начертить контур детали при помощи идущих друг за другом отрезков, причём следующий отрезок начинается от конца

предыдущего (используется команда Отрезок); и наконец, если деталь симметрична, то можно построить половину вида, а затем просто зеркально его отобразить, получив тем самым интересующий нас вид.

Начертим вспомогательные линии, относительно которых будут проводиться дальнейшие построения (рис. 6.64).

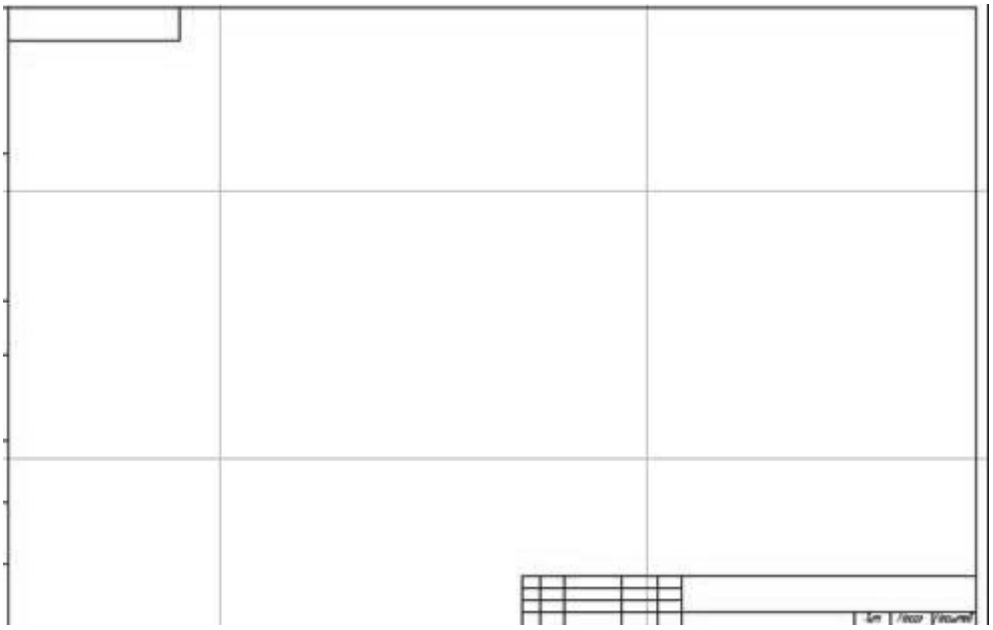


Рисунок 6.64 – Вычерчивание самой детали

Сперва начертим половину вида (рис. 6.65), привязанную к осевой линии.

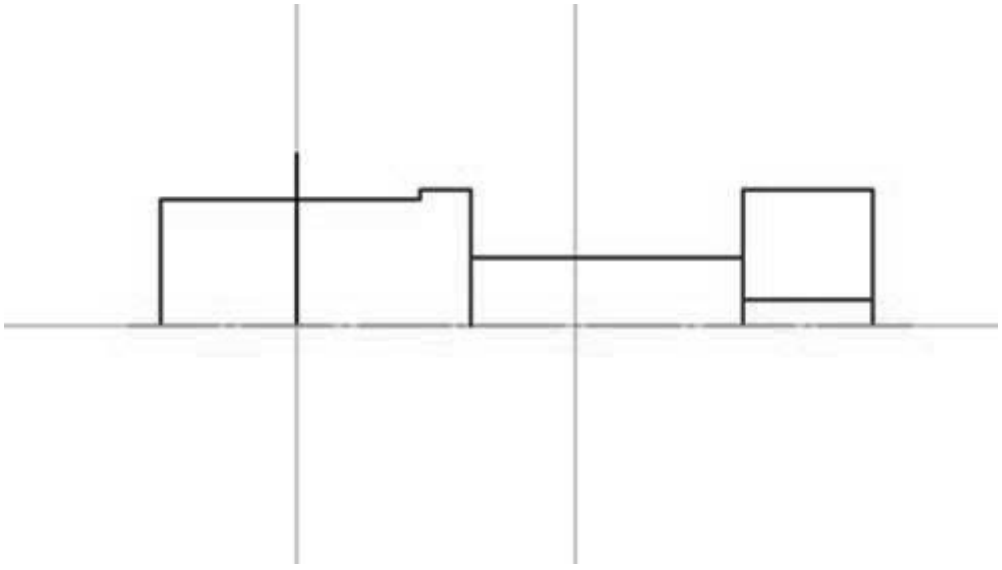


Рисунок 6.65 — Половина вида

Дальше с помощью команды Зеркало (рис. 6.66) отобразим вторую половину вида относительно осевой линии, причём исходный объект был сохранён на старом месте.

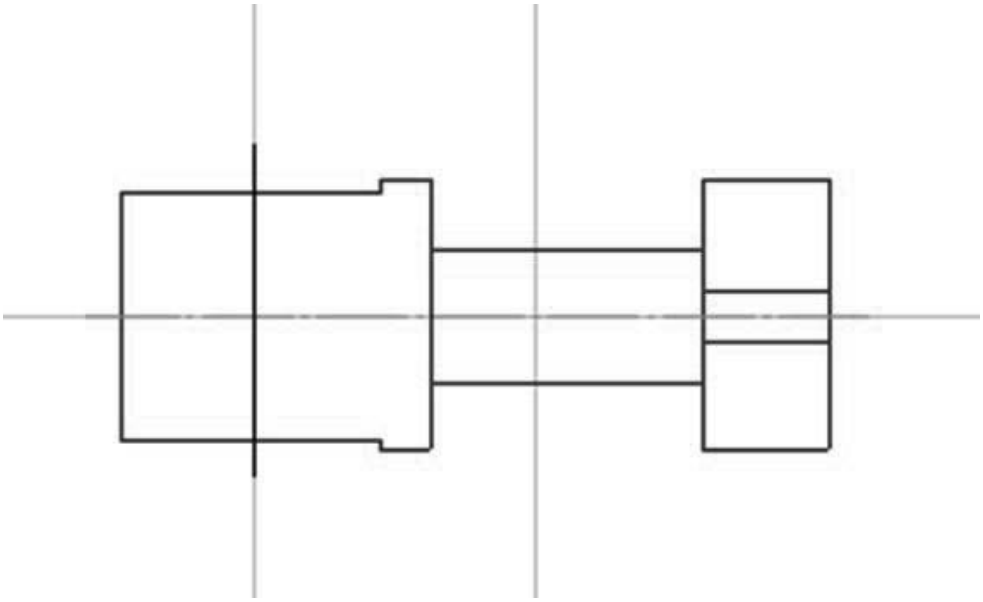


Рисунок 6.66 — Команда Зеркало

После этого достроим недостающие объекты детали с помощью команды Отрезок (рис. 6.67).

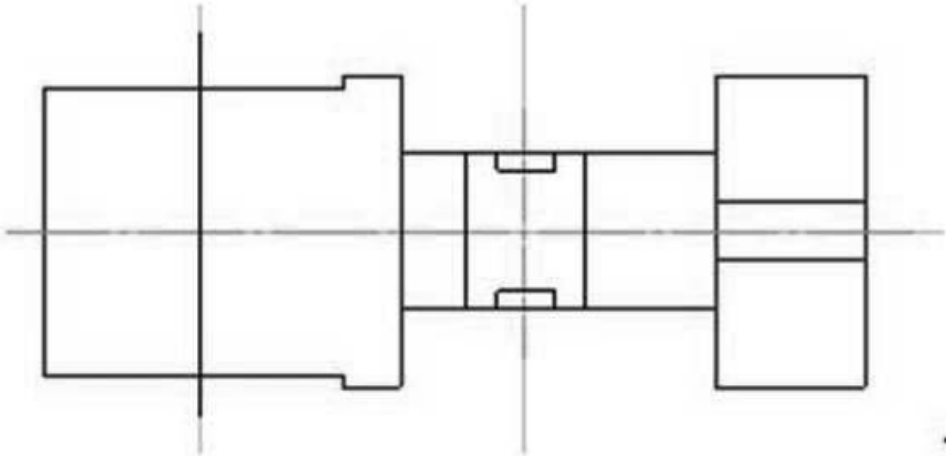


Рисунок 6.67 — Команда Отрезок

Вычерчивание сечений детали (рис. 6.68). Сечения вычерчиваем на всё тех же осевых линиях, естественно создав предварительно слои Вынесенные сечения и Наложенное сечение (тип линий – сплошная, тощина линий – 1 мм). Сечение прямоугольной формы строим следующим образом:

1. Начертим окружность с центром в точке пересечения осевых линий и заданным радиусом.
2. На заданном расстоянии от центра построим по паре прямых, параллельных осям OX и OY.

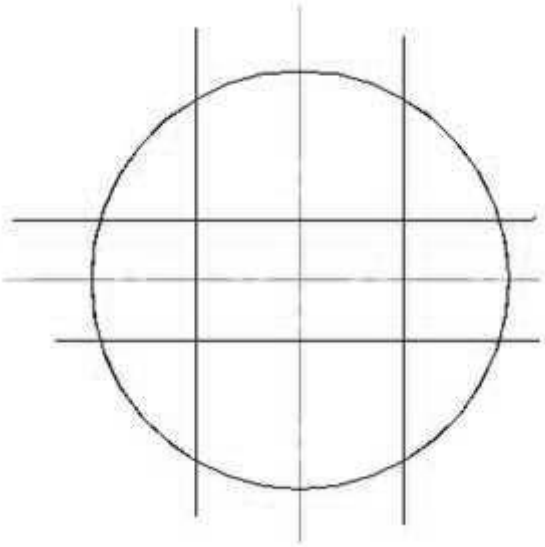


Рисунок 6.68 — Вычерчивание сечений детали

С помощью команды Усечь кривую отделим ненужные части объектов и удалим их (рис. 6.69).

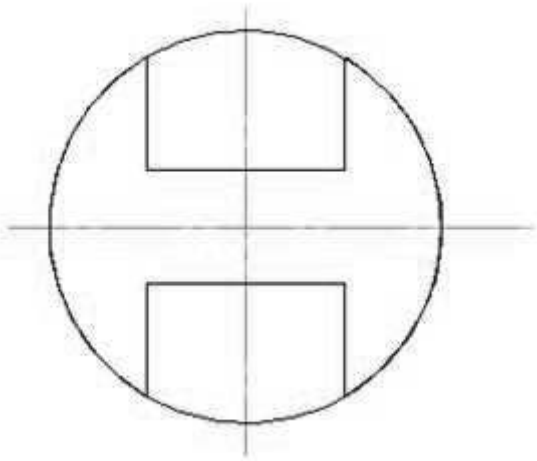


Рисунок 6.69

Второе вынесенное сечение начертим, используя такой же метод. Сначала построим окружность с центром в точке пересечения осевых линий и заданным радиусом. Достроим по 4 параллельные линии OX и OY (рис. 6.70).

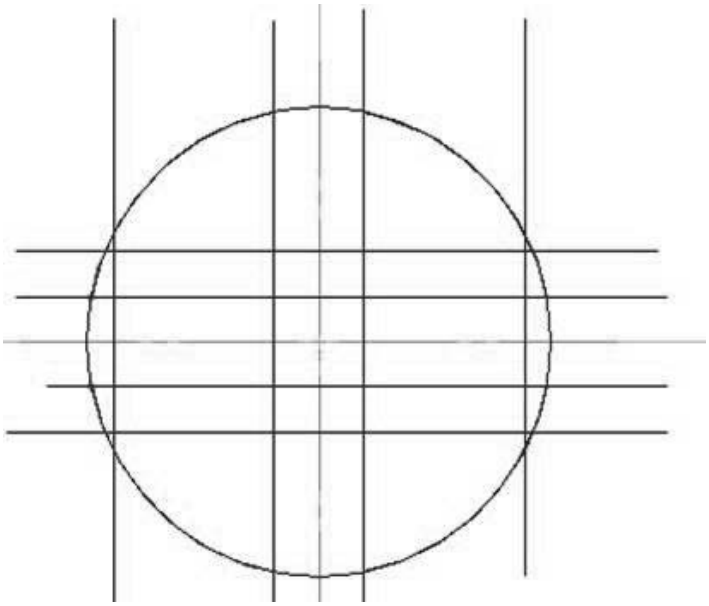


Рисунок 6.70

С помощью команды Усечь кривую отделим ненужные части объектов и удалим их (рис. 6.71).

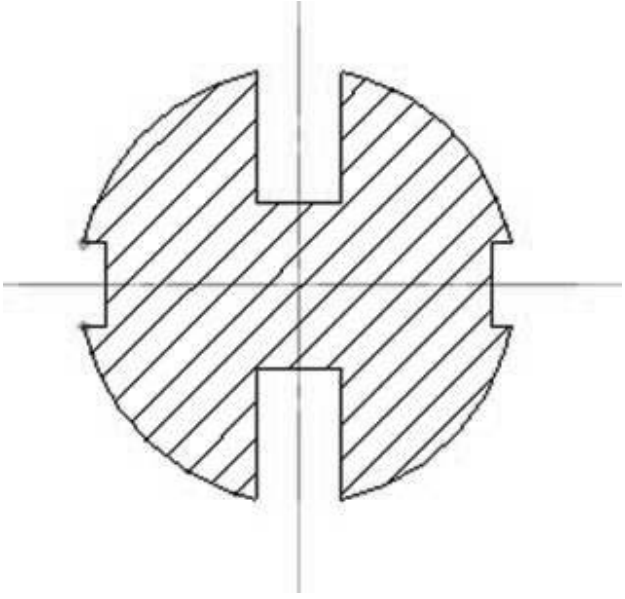


Рисунок 6.71

Для наложенного сечения создадим слой Наложенное сечение (тип линий – сплошная, толщина линий – 0,5 мм) и начертим его с помощью двух параллельных отрезков (рис. 6.72).

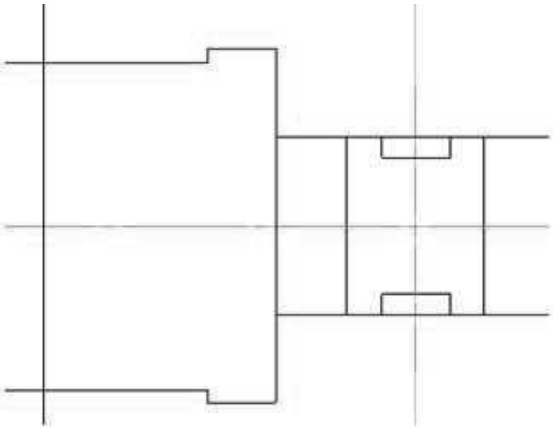


Рисунок 6.72

После всего этого поставим штрихи, обозначающие секущую плоскость, стрелки, обозначающие направление взгляда, и обозначим сечения большими буквами А.

Нанесение на сечения детали штриховки (рис. 6.73)

Штриховку сечений выполняем с помощью диалогового окна Штриховка (Инструменты\ Штриховка…). Для этого:

1. Вызовем диалоговое окно Штриховка.

2. На появившейся панели настроек выберем стиль и цвет штриховки (а также можно задать шаг и угол) .

3. Далее после щелчка на кнопке Указание точек выберем области, которые нужно заштриховать.

После предварительного просмотра результата подтвердим завершение выполнения команды.

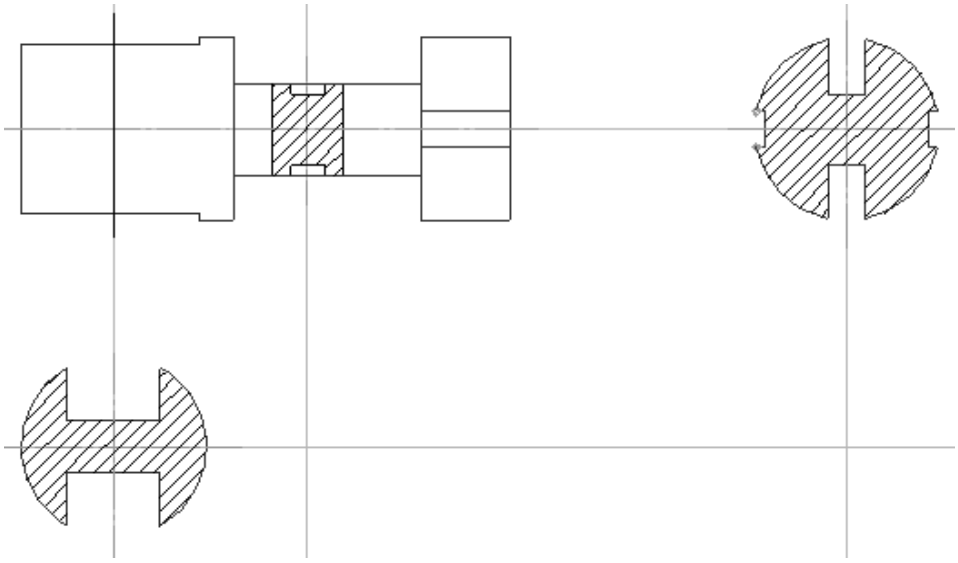


Рисунок 6.73 — Нанесение на сечения детали штриховки

Простановка размеров ничем особенным не отличается. Все размеры – параллельные. При их простановке, при помощи мыши, указываются: начальная точка первой выносной линии, начальная точка второй выносной линии и расстояние от размерной линии до контура детали. Единственное, на что следует обратить внимание, так это простановка специальных символов, таких как: диаметр и квадрат. Чтобы поставить знак диаметра перед размерным числом, необходимо:

1. Указать начальную точку первой выносной линии.
2. Указать начальную точку второй выносной линии.
3. В окне встроенного текстового редактора щёлкнуть ПКМ на чистой области и из выпадающего меню выбрать Символ \ Диаметр.
4. Нажать кнопку <Ok>.
5. Указать расстояние от размерной линии до контура детали.

Финальный вид чертежа (рис. 6.74)

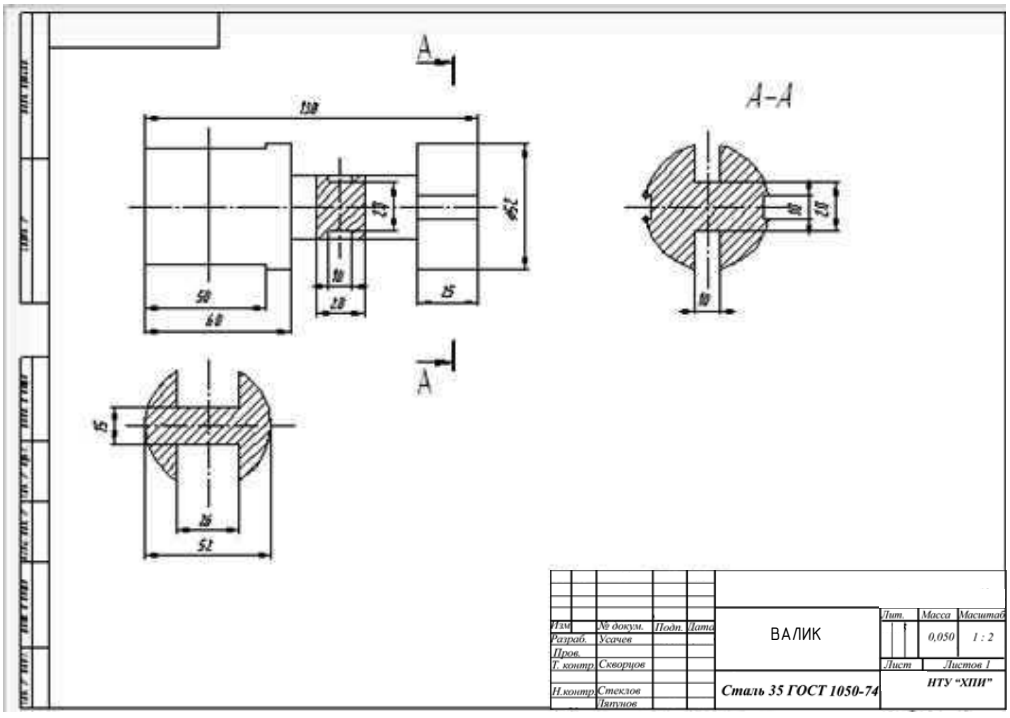


Рисунок 6.74 — Финальный вид чертежа

6.7. **Конструирование объектов с использованием пакета** Illustrator

Конструирование — деятельность по созданию материального образа разрабатываемого объекта, ему свойственна работа с натурными моделями и их графическими изображениями (чертежи, эскизы, компьютерные модели). Эти модели и изображения, а также некоторые виды изделий называют конструкциями.

Конструирование может осуществляться:

- вручную при помощи чертёжных инструментов;
- при помощи графических редакторов;
- автоматизированно — при помощи систем автоматизации проектных работ (САПР);

- автоматически (без участия человека) при помощи Интеллектуальной информационной системы (ИИС)

Графический редактор — программа (или пакет программ), позволяющая создавать и редактировать изображения с помощью компьютера. Существуют растровые и векторные редакторы (рис. 6.75).

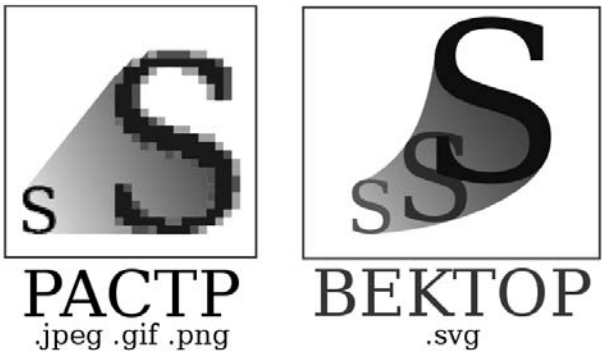


Рисунок 6.75 — Растровые и векторные редакторы

Растровый графический редактор — специализированная программа, предназначенная для создания и обработки растровых изображений, которые, в свою очередь, представляют собой сетку пикселей или цветных точек на мониторе, бумаге и других отображающих устройствах и материалах (растр). Создается растровая графика фотоаппаратами, сканерами, непосредственно в растровом редакторе, также путем экспорта из векторного редактора или в виде снимка экрана.

Важными характеристиками изображения являются:

- количество пикселей — размер;
- количество используемых цветов или глубина цвета;
- цветовое пространство (цветовая модель) RGB, CMYK, LAB и др.;
- разрешение — справочная величина, говорящая о рекомендуемом размере пикселя изображения.

Растровые редакторы больше подходят для обработки и ретуширования фотографий, создания фотореалистичных иллюстраций и коллажей.

Векторная графика — способ представления объектов и изображений в компьютерной графике, основанный на использовании элементарных геометрических объектов, таких как точки, линии, сплайны и многоугольники. Объекты векторной графики являются графическими изображениями математических функций. Векторные графические редакторы позволяют пользователю создавать и редактировать векторные изображения непосредственно на экране компьютера. Первые обычно более пригодны для создания разметки страниц, типографики, логотипов, sharp-edged artistic иллюстраций (например, мультипликация, clip art, сложные геометрические шаблоны), технических иллюстраций, создания диаграмм и составления блок-схем.

Основные инструменты векторных редакторов:

- кривые Безье — позволяют создавать прямые, ломаные и гладкие кривые, проходящие через узловые точки, с определёнными касательными в этих точках;
- заливка — позволяет закрашивать ограниченные области определённым цветом или градиентом;
- текст создаётся с помощью соответствующего инструмента, а потом часто преобразуется в кривые, чтобы обеспечить независимость изображения от шрифтов, имеющихся (или отсутствующих) на компьютере, используемом для просмотра;

- набор геометрических примитивов;
- карандаш — позволяет создавать линии «от руки». При создании таких линий возникает большое количество узловых точек, от которых в дальнейшем можно избавиться с помощью «упрощения кривой».

Среди векторных графических редакторов наиболее популярны *Corel Draw* и *Adobe Illustrator*.

Главенствующее место по праву занимает *Adobe Illustrator*. Достаточно сказать, что *Illustrator* — фактически мировой стандарт для векторных работ полиграфической направленности. С недавних пор Иллюстратор активно предпринимает попытки занять позиции и в области подготовки web-иллюстраций. Однако его основным непревзойденным свойством пока по-прежнему остается работа без новомодной приставки e-. Это создание векторной графики, предназначенной для переноса на так называемый «твердый» носитель, на бумагу. Все инструменты для создания высококачественного рисунка и подготовки его к печати лежат «на поверхности» интерфейса. Надо учесть и то, что программа принадлежит фирме Adobe — создателю языка PostScript, что предопределяет максимально возможную совместимость программы с этим важнейшим промышленным стандартом. Одной из главных черт *Adobe Illustrator* является его корректность в работе с вектором. То, что сделано в Иллюстраторе, почти наверняка будет правильно выведено на фотонаборном автомате. Важно, что элемент случайности практически исключен.

6.7.1. ***Перспектива***

При конструировании разрабатываемого объекта применяется один из основных методов построения изображений на плоскости – метод перспективных проекций. Такие изображения являются наиболее наглядными и позволяют передавать предметы, как существующие, так и не существующие: проектируемые. Перспектива позволяет не только представить будущее изделие, но и своевременно выявить достоинства или недостатки формы, композиционного или цветового решения проекта. С ее помощью удобно проверить и корректировать решения. Во многих случаях перспективные изображения успешно заменяют макеты сложных по форме и цветовым решениям объектов. Высокие иллюстративные свойства перспективных изображений делают их незаменимыми в творческом процессе.

Время возникновения первых перспективных изображений точно не установлено. С тех пор как люди стали изображать окружающий их видимый мир, они путем наблюдения натуры постепенно постигали геометрические свойства перспективных проекций. Процесс этот был длительным и не мог проходить одновременно и одинаково во всех уголках земного шара. Поэтому сроки возникновения первых перспективных изображений, определенные археологическими исследованиями, могут подвергнуться пересмотру в свете новых данных позднейших раскопок.

Одной из первых известных научных работ по перспективе был труд Эвклида (III в. до н. э.), названный «Оптика», в котором содержится 61 теорема и 12 аксиом. До нас дошли еще более древние сочинения о перспективе Птолемея. Много задач по построению перспективных изображений рассматривал римский архитектор Витрувий в своем труде «Десять книг об архитектуре» (I в. до н. э.), театральные декорации греческого художника Агафарга (вторая половина V в. до н. э.), росписи домов художников Помпеи – в основном «фронтальная» перспектива (I в. до н. э. – I в н. э.) и др.

Изучение фресок и мозаик древней Руси приводит к выводу, что уже в X-XII веках русские художники-иконописцы были знакомы с наблюдательной перспективой. И в допетровское время в России стали достаточно грамотно выполнять изображения методами центрального и параллельного проектирования.

Все древние авторы характеризуют перспективные и близкие к ним изображения словами: искусство правильно видеть (perspettiva - от глагола на итальянском языке «perspicere» — правильно,

хорошо видеть, от латинского слова «perspicio» — «ясно вижу», и французского — «laperspective» — «вид вдаль»). Введение ряда терминов перспективы относится к эпохе Возрождения: центр проецирования, картинная плоскость, линия горизонта и т. д. С позиций теории, и особенно практики, перспектива являлась и продолжает оставаться сложным инструментом для освоения и применения. Однако современный инструментарий в виде персональных компьютеров и систем диалогового моделирования позволяет весь, практически необозримый материал по перспективе, свести к небольшому числу операций. Методы построения перспективы:

1. Радиальный метод (следа луча).

Сущность данного метода, разработанного Дюрером (1471-1528), заключается в том, что картинная плоскость занимает либо фронтальное положение в ортогональных проекциях, либо профильное, а перспектива точки пространства определяется как картинный след луча зрения, проходящего через эту точку.

На приведенном примере показано построение перспективы точки А (рис. 6.76).

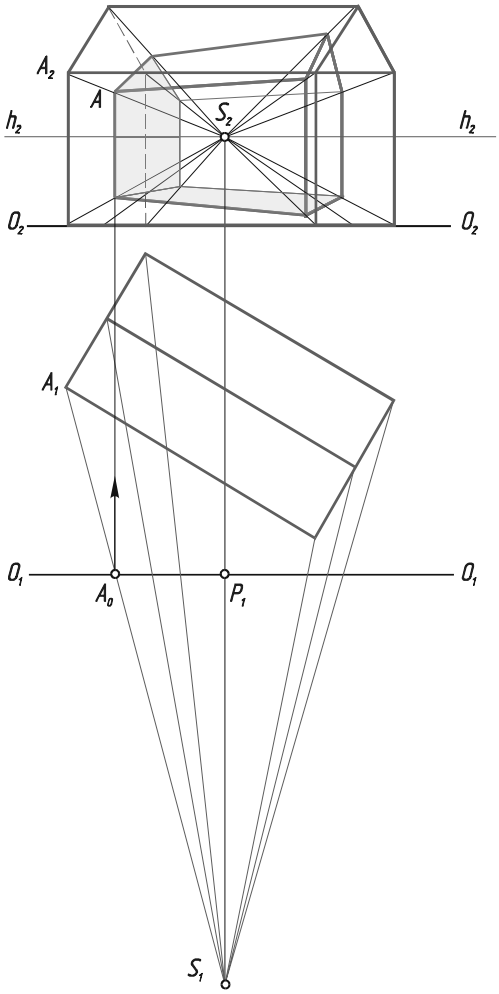


Рисунок 6.76 — Построение перспективы точки *А*

Положительной стороной метода является простота теории, позволяющая без знания теоретических основ линейной перспективы осуществлять перспективное проецирование. Отрицательной стороной является загроможденность изображения линиями вспомогательных построений, наложение перспективного изображения на ортогональный чертеж и др.

2. Метод архитекторов.

В практике работы архитектурных мастерских широко применяется метод построения перспективных изображений с использованием точек схода параллельных прямых.

Построение перспективы данным методом основано на использовании ортогональных проекций предмета и может осуществляться на отдельном листе. Сущность метода сводится к построению перспективы основания (плана) предмета и к последующему определению положения отдельных точек изображения по высоте (рис. 6.77).

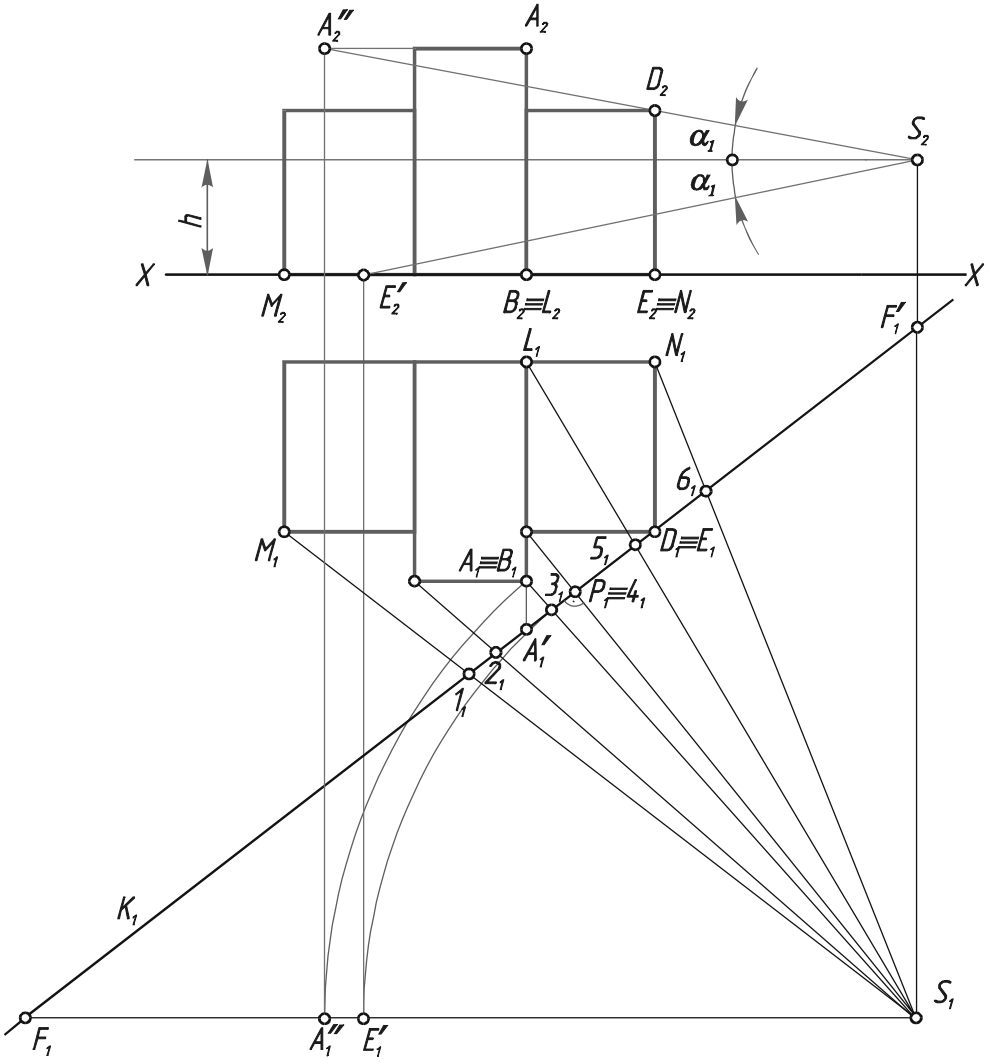


Рисунок 6.77 – Построение перспективы методом архитектора

Построение проводим в следующем порядке.

1. Руководствуясь вышеизложенными правилами назначения точки зрения и картины, через ребро $D_1 \equiv E_1$ плана тела проводим след $K_1(0_1 0_1)$ картинной плоскости, намечаем основания точки зрения S_1 (точку зрения) и главной точки P_1. Проводим линию горизонта на расстоянии h от линии основания картины (на фронтальной проекции).

2. На горизонтальной проекции (рис. 6.78) проводим прямые, соединяющие основание точки зрения S_1 со всеми видимыми вершинами основания предмета.Точки пересечения *1, 2, 3, 4, 5* и *6* этих прямых с основанием картины переносим в перспективу и проводим через них тонкие вертикальные линии. Переносим в перспективу также точки D_1 E_1 и A_1'.

3. Проводим на горизонтальной проекции прямые S_1F_1 и S_1F_1' (проекции лучей *SF* и *SF'*), параллельные сторонам основания предмета, до пересечения с основанием картины K_1 в точках F_1 и F_1' (горизонтальные проекции точек схода); определяем (рис. 6.78) на линии горизонта точки схода *F* и *F'* горизонтальных ребер данного тела.

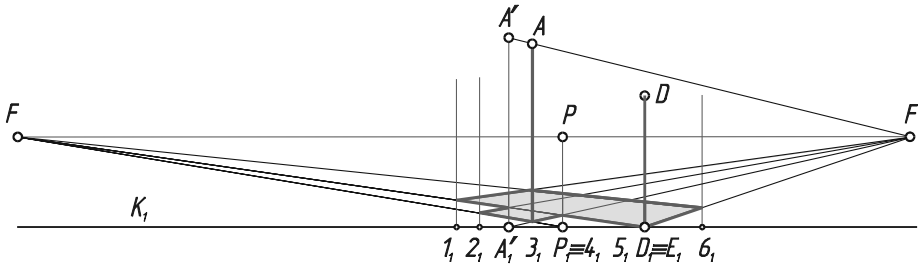


Рисунок 6.78 — Перспектива сторон

4. Строим перспективу основания тела. Прежде всего, строим перспективу сторон *EN, EM* и *BL*, соединяя точку $D_1 \equiv E_1$ (рис. 6.91) с точками *F* и *F'*, а точку *A'* — с точкой *F'*,и определяем положение вершин *N, M, B* и *L* (в перспективе они не обозначены) при помощи вертикальных прямых, проходящих через точки *1, 3, 5* и *6*.

5. Имея перспективу вершин *E, N, M* и *B*, строим перспективу остальных сторон и вершин основания. Строим перспективу ребра *DE* или (DD_1), откладывая от точки D_1 вверх вдоль вертикальной прямой натуральную длину этого ребра.

6. Строим перспективу ребра *AB*, откладывая его натуральную длину в виде отрезка $A_1'A'$, расположенного в картинной плоскости. Отрезок $A_1'A'$ можно рассматривать или как проекцию ребра *AB* на плоскости *K* (рис. 6.92), или как линию пересечения с плоскостью картины продолженной грани предмета.

7. Дальнейшие построения выполнены на рис.6.79а и заключаются в проведении горизонтальных ребер предмета, идущих в точки схода *F* и *F'*.

Примечание. Перспектива тела построена в масштабе 2:1 по отношению к размерам ортогонального чертежа.

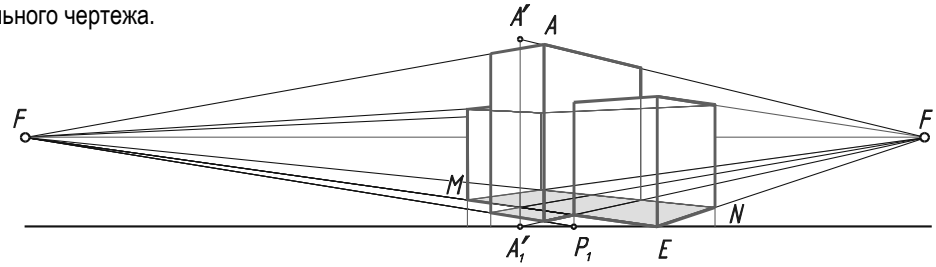


Рисунок 6.78 а

Способ перспективной сетки предложил в XV в. итальянский зодчий Альберти. Суть способа состоит в следующем (рис. 6.79).

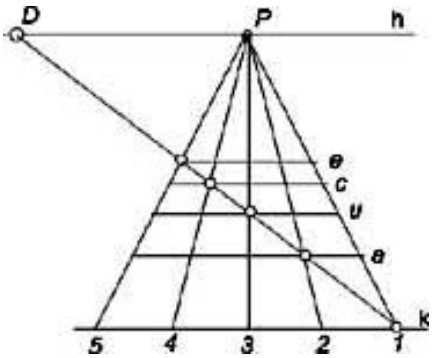


Рисунок 6.79

Соединив точку схода *P* с точками на следе картинной плоскости *1, 2, 3,* …, получим перспективные изображения первого семейства прямых, перпендикулярных к картине. Далее через точку *1* проводим линию *1-D. D* – точка схода (другое название дистанционная точка) любых горизонтальных прямых, наклоненных к картине под тем или иным углом (в данном случае 45 градусов). Перспективные изображения прямых *a, b, c,* …пройдут через соответствующие точки пересечения прямой *1-D* с линиями *P – 2, P – 3, P – 4,* … .Далее, чтобы начертить кривую, или какой орнамент с ортогональной сетки плана на перспективную сетку, используется художественный прием рисования «по клеткам».

Удобный и часто используемый в практической перспективе метод боковой стенки (рис.6.80) был введен в 1693 г. итальянским художником Андреа Поццо. Прием состоит в следующем. На свободном месте картины, сбоку, зафиксировав на линии горизонта h произвольную точку D, до следа картинной плоскости k произвольную прямую OD. Из точки O восстановить перпендикуляр («масштабный шест») к основанию картины k. Используя «шест», откладываем от точки O нужную величину (в масштабекартины), и соединяяее с D получим изменения данной величины вглубь.

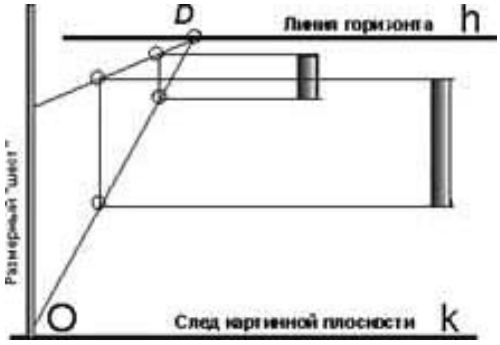


Рисунок 6.80

6.7.1.1. ***Пример построение перспективы в*** *Adobe Illustrator.*

Этапы построения прямоугольника и заливка его плиткой представлены на рис. 6.81, 6.82, 6.83.

Рисунок 6.81

Выделим прямоугольник и нажмем Object/Expand. Возьмем инструмент Free Transform Tool(рис. 6 .81), потянем за левый верхний угол вправо и нажмем Ctrl, продолжая перемещать мышь к центру прямоугольника.

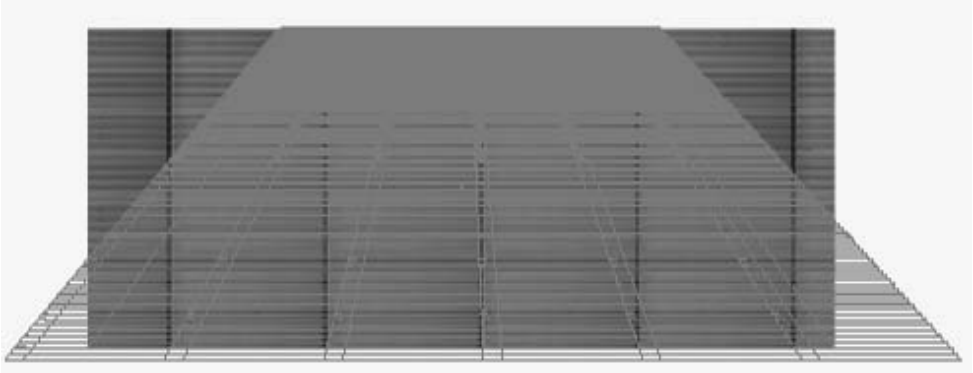


Рисунок 6.82

Перспектива уходящей вдаль дороги готова (рис. 6.83).

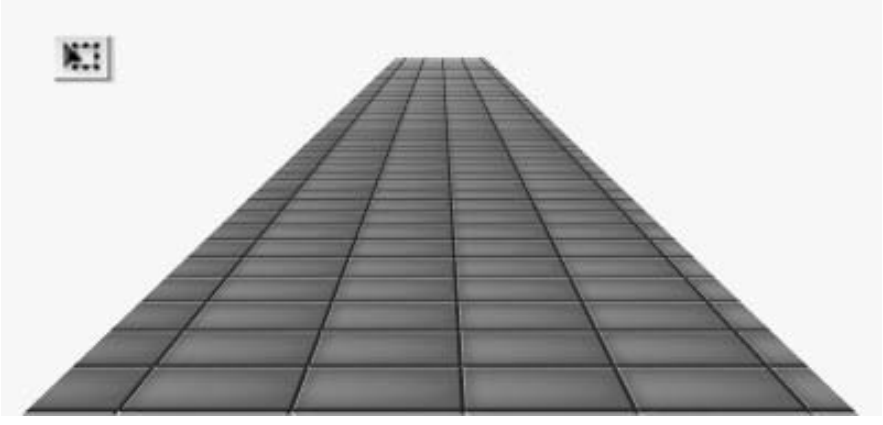


Рисунок 6.83

Также в Illustrator предоставляются стандартные стили одно-, двух- и трехточечных перспектив (рис. 6.84).

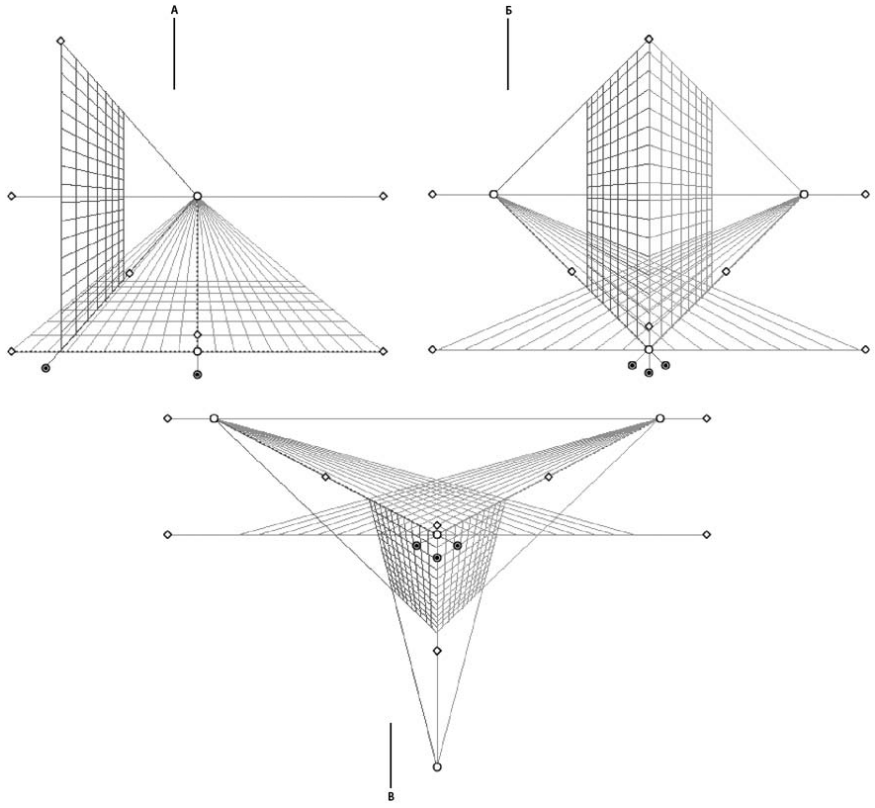


Рисунок 6.84 — Стандартные стили одно-, двух- и трехточечных перспектив.

Стили сетки перспективы А. 1-точечная перспектива Б. 2-точечная перспектива (по умолчанию) В. 3-точечная перспектива

Чтобы выбрать один из стандартных стилей сетки перспективы, выберите «Просмотр» > «Сетка перспективы», а затем — требуемый стиль (рис. 6 .85).

Чтобы определить настройки сетки, выберите «Просмотр» > «Сетка перспективы» > «Определить сетку» — в диалоговом окне «Определить сетку перспективы». Новые объекты в перспективе рисуются с помощью инструментов групп линий или прямоугольников при отображении сетки. При использовании инструментов групп прямоугольников или линий можно переключиться на инструмент «Выбор перспективы. Можно также переключить активную плоскость, нажимая «горячие» клавиши 1 (левая плоскость), 2 (горизонтальная плоскость) и 3 (правая плоскость), с одновременным выбором этих инструментов.

Когда создается объект в перспективе, используются «быстрые» направляющие, чтобы выровнять объект относительно других объектов. Выравнивание основано на геометрии перспективы объектов. Направляющие отображаются при приближении объекта к краю или опорной точке других объектов.

Параметры сетки перспективы можно настроить в окнах «Просмотр» > «Сетка перспективы».

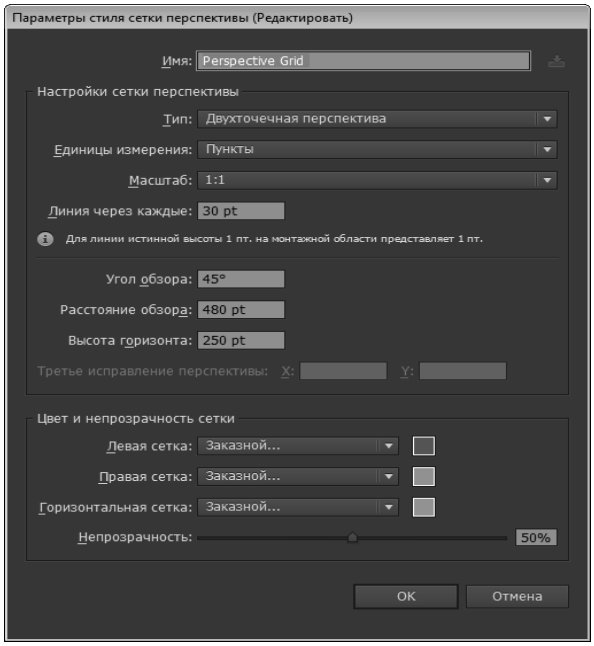


Рисунок 6.85

6.8. **Конструирование объектов с использованием пакета** Adobe Photoshop

Adobe Photoshop — многофункциональный графический редактор, разработанный и распространяемый фирмой Adobe Systems. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты. Продукт является лидером рынка в области коммерческих средств редактирования растровых изображений, и наиболее известным продуктом фирмы Adobe. Часто эту программу называют просто *Photoshop*. В настоящее время Photoshop доступен на платформах OS X, Windows, в мобильных системах iOS и Android.

Несмотря на то, что изначально программа была разработана как редактор изображений для полиграфии, в данное время она широко используется и в веб-дизайне. В более ранней версии была включена специальная программа для этих целей — Adobe ImageReady, которая была исключена из версии CS3 за счёт интеграции её функций в сам *Photoshop*, а также включения в линейку программных продуктов Adobe Fireworks, перешедшего в собственность Adobe после приобретения компании Macromedia.

Photoshop тесно связан с другими программами для обработки медиафайлов, анимации и другого творчества. Совместно с такими программами, как Adobe ImageReady (программа упразднена в версии CS3), Adobe Illustrator, Adobe Premiere, Adobe After Effects и Adobe Encore DVD enru, он может использоваться для создания профессиональных DVD, обеспечивает средства нелинейного монтажа и создания таких спецэффектов, как фоны, текстуры и т. д. для телевидения, кинематографа и всемирной паутины. Photoshop также прижился в кругах разработчиков компьютерных игр.

Основной формат Photoshop, PSD, может быть экспортирован и импортирован всеми программными продуктами, перечисленными выше. Photoshop CS поддерживает создание меню для DVD. Совместно с Adobe Encore DVD, Photoshop позволяет создавать меню или кнопки DVD. Photoshop CS3 в версии Extended поддерживает также работу с трёхмерными слоями.

Из-за высокой популярности Photoshop поддержка специфического для неё формата PSD была реализована во многих графических программах, таких как Adobe Fireworks, Photo-Paint, WinImages enru, GIMP, PaintShop Pro и других.

Photoshop поддерживает следующие цветовые модели или способы описания цветов изображения (в нотации самой программы — режим изображения): RGB, LAB, CMYK, В градациях серого, Черно-белые, Duotone enru, С 256-цветовой палитрой (Indexed), Многоканальные (Multichannel).

Поддерживается обработка изображений, с глубиной цвета 8 бит (256 градаций на один канал), 16 бит (используется 15 битов плюс один уровень, то есть 32769 уровней) и 32 бит (используются числа одинарной точности с плавающей запятой). Возможно сохранение в файле дополнительных элементов, как то: направляющих (Guide), каналов (например, канала прозрачности — Alpha channel), путей обтравки (Clipping path), слоёв, содержащих векторные и текстовые объекты. Файл может включать цветовые профили (ICC), функции преобразования цвета (transfer functions). Допускаются неквадратные пиксели (Pixel Aspect Ratio).

*6.8.1. **История версий** Adobe Photoshop*

Первая версия появилась в 1987 году. Её создал студент Мичиганского университета Томас Нолл для платформы Macintosh. Он назвал её Display, но в 1988 году переименовал в ImagePro. В сентябре 1988 года Adobe Systems купила права на программу, оставив разработчиком Томаса Нолла, а в 1989 году программу переименовали в *Photoshop*. В 1990 году появился *Photoshop 1.0*.

Photoshop 8.0, датируемый октябрём 2003 года, имеет название *Photoshop CS*, так как начал относиться к новой линейке продуктов компании Adobe Systems — Creative Suite.

Photoshop 10.0, датируемый апрелем 2007 года, имеет название Photoshop CS3. Аббревиатура CS3 означает, что продукт интегрирован в третью версию пакета программ Adobe Creative Suite. В предыдущих продуктах — *Photoshop CS* и *CS2*, с целью отличия от прежних версий и укрепления принадлежности к новой линейке продуктов, был изменён символ программы: вместо изображения глаза, которое присутствовало в версиях с 3-й по 7-ю, в стилевом решении использовалось изображение перьев. В *Photoshop CS3* в иконке приложения и экране-заставке используются буквы из названия продукта «Ps» на синем градиентном фоне рис. 6.86.

Рисунок 6 .86 - Официальное лого

Список нововведений включает в себя новый интерфейс, увеличенную скорость работы, новый Adobe Bridge, новые фильтры и инструменты, а также приложение Adobe Device Central enru, позволяющее осуществлять предварительный просмотр работы в шаблонах популярных устройств, например мобильных телефонов.

Photoshop 14, датируемый июнем 2013 года, имеет название *Photoshop CC*. Аббревиатура CC означает, что продукт интегрирован в пакет программ Adobe Creative Cloud.

Начиная с июня 2014 года программа имеет новое именование версий: теперь она называется не *Photoshop 15.0*, как должно быть, или *Photoshop CC2*, как это было во время существования программы в Adobe Creative Suite, а *Photoshop 2014.0.0*, указывая тем самым на год выпуска.

Последние версии включают в себя Adobe Camera RAW — плагин, разработанный Томасом Ноллом, который позволяет читать ряд Raw-форматов различных цифровых камер и импортировать их напрямую в *Photoshop*.

Расширенная версия программы *Adobe Photoshop Extended* предназначена для более профессионального использования, а именно — при создании фильмов, видео, мультимедийных проектов, трехмерного графического дизайна и веб-дизайна, для работы в областях производства, медицины, архитектуры, при проведении научных исследований.

В программе *Adobe Photoshop Extended* современных версий (начиная с CS4) можно открывать

и работать с 3D-файлами, создаваемыми такими программами, как Adobe Acrobat 3D, Autodesk 3ds Max, Maya и Google Планета Земля. Photoshop поддерживает следующие форматы файлов 3D: U3D, 3DS, OBJ, KML и DAE.

Возможно использовать трехмерные файлы для внедрения в двумхмерное фото. Доступны некоторые операции для обработки 3D-модели, такие как работа с каркасами, выбор материалов из текстурных карт, настройка света. Также можно создавать надписи на 3D-объекте, вращать модели, изменять их размер и положение в пространстве. Программа включает в себя также команды по преобразованию плоских фотографий в трехмерные объекты определенной формы, такие как, например, банка, пирамида, цилиндр, сфера, конус и др.

Для имитации движения в *Photoshop* можно создавать кадры мультипликации, используя слои изображения. Можно создавать видеоизображения, основанные на одной из многих заданных пиксельных пропорций. После редактирования можно сохранить свою работу в виде файла GIF-анимации или PSD, который впоследствии можно проиграть во многих видеопрограммах, таких как Adobe Premiere Pro или Adobe After Effects. Доступно открытие или импортирование видеофайлов и последовательности изображений для редактирования и ретуширования, создание видеоряда мультипликации и экспорт работ в файл формата QuickTime, GIF-анимацию или последовательность изображений. Видеокадры можно отдельно редактировать, трансформировать, клонировать, применять к ним маски, фильтры, разные способы наложения пикселов, на них можно рисовать, используя различные инструменты.

Начиная с версии *CS* в *Photoshop* доступна работа со скриптами.

Photoshop поддерживает файлы DICOM (Digital Imaging and Communications in Medicine) — цифровые изображения и коммуникации в медицине. Для открытого в *Photoshop* DICOM-файла, можно использовать любой инструмент *Photoshop* для коррекции и ретуширования изображений.

И, наконец, с помощью программы *Photoshop Extended* можно рассматривать *MATLAB*-изображения, обрабатывать их в программе *Photoshop*, комбинировать команды *MATLAB* с технологиями обработки изображений *Photoshop*. Как только устанавливается соединение с программой *Photoshop* из программы *MATLAB* и осуществляется ввод команд в командную строку *MATLAB*, эти управляющие воздействия незамедлительно выполняются в *Photoshop*. Файлы, подготовленные в программе *MATLAB*, имеют расширение m, fig, rpt, mat, mdl. Коммуникация между *Photoshop* и *MATLAB* использует интерфейс *Photoshop JavaScript* и библиотечный интерфейс *MATLAB*.

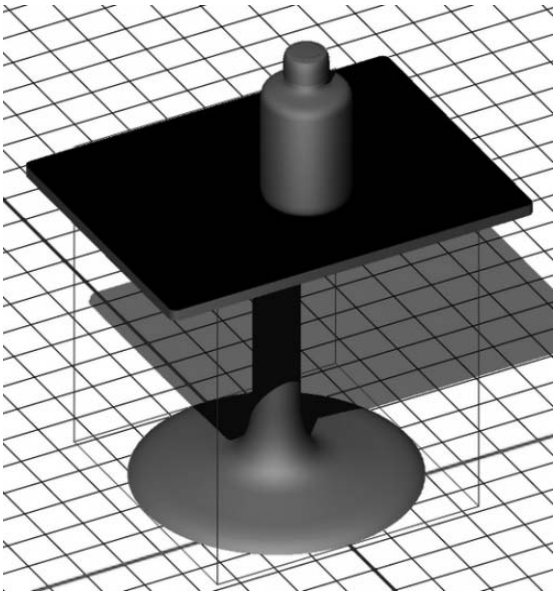


Рисунок 6.87 – Пример 3D –моделирования

*6.8.2. **Пример моделирования в** Adobe Photoshop CS6*

Пример 3D-моделирования показан на рис. 6.87.

Создается новый файл в *Photoshop*. Первое, что создается, это стол. Берется ручка Pen Tool (P) и рисуется фигура, как показано на изображении ниже (рис. 6.88).

Включается режим 3D>New Extrusion from Selected Layer, и выставляются следующие параметры (рис. 6.89).

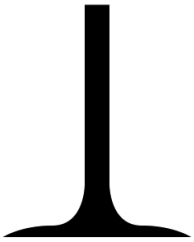


Рисунок 6.88 – ножка стола

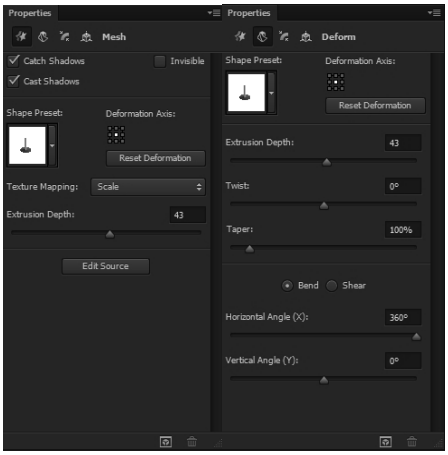


Рисунок 6.89 - Параметры

Далее создается большой прямоугольник с закругленными краями. Затемменю 3D>New Extrusion from Selected Layer (or Path). Выставляются настройки (рис. 6.90).

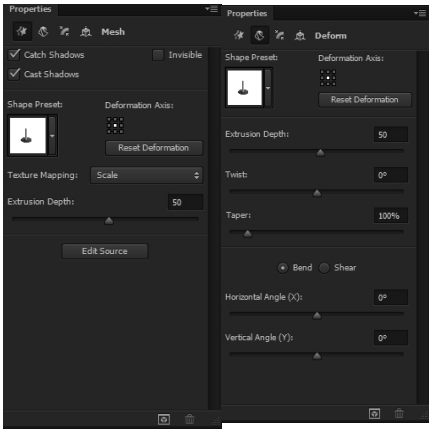


Рисунок 6.90 — Параметры

На панели слоев выбираются созданные фигуры и затем меняется режим в меню 3D>Merge 3D Layers. Стол готов (рис. 6.91).

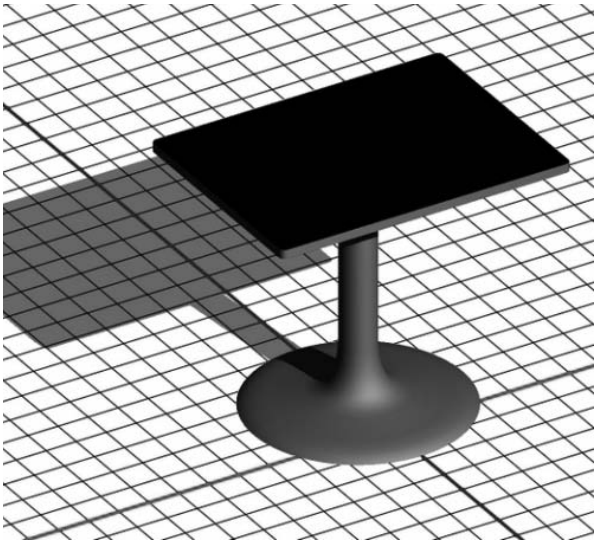


Рисунок 6.91 — Стол

Создается бутылка. При помощи ручки рисуются элементы (рис. 6.92).

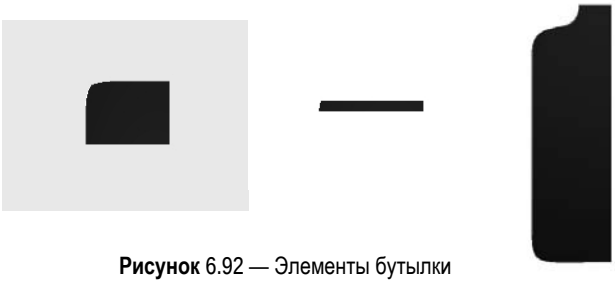


Рисунок 6.92 — Элементы бутылки

На панели слоев выбираются 3 созданные фигуры. Переход в меню 3D>Merge 3D Layers. Результат (рис. 6.93):

Рисунок 6.93 — Бутылка в 3D

Конечный результат можно посмотреть на рисунке 6.94

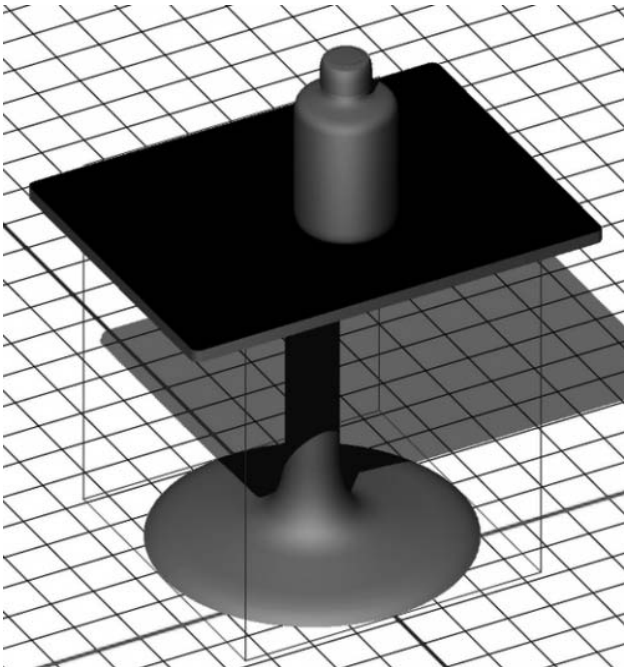


Рисунок 6.94– Конечный результат

ВЫВОДЫ

Программы векторной графики работают с объектами, которые могут быть созданы на основе кривых и геометрических фигур и сохранены в памяти компьютера в виде описаний контуров. Это различные схемы, логотипы, пиктограммы, рисунки, текстовые объекты. Ими пользуются как художники и дизайнеры, так и люди других профессий при подготовке файлов технической документации, описании схем, планов, чертежей, оформлении курсовых и дипломных работ, рефератов и т.д. Главное достоинство векторных файлов по сравнению с растровыми – меньший размер и удобство редактирования, но при их выводе на экран производится множество математических операций. Поэтому скорость вывода векторных изображений обычно меньше, чем растровых, хотя этот недостаток довольно эффективно устраняется при помощи специальных процессоров – графических ускорителей. Существует целый ряд программ, переводящих графические данные из векторного формата в растровый. Но обратная задача (перевод растровых изображений в векторные) является довольно сложной и решается только в наиболее совершенных графических пакетах. Не менее сложные проблемы возникают и при преобразованиях одного векторного формата в другой, так как многие графические пакеты используют уникальные математические модели для элементов изображения.

Базовым продуктом среды 3D-моделирования является, как правило, математическая модель твердого тела: обладающая топологией, геометрией, набором физико-механических свойств, необходимых для анализа поведения деталей и сборочных единиц и обеспечения их работоспособности на этапе проектирования.

В некоторых системах 3D имеются средства автоматического анализа физических характеристик, таких как вес, моменты инерции и средства решения геометрических проблем сложных сопряжений и интерпретации. Поскольку в 3D системах существует автоматическая связь между данными различных геометрических видов изображения, 3D моделирование полезно в тех приложениях, где требуется многократное редактирование 3D образа на всех этапах процесса проектировании.

К недостаткам 3D-графики можно отнести:

- высокие требования к аппаратной составляющей компьютера: к его оперативной памяти, быстроте работы процессора и т.д.;
- необходимость больших временных затрат на создание моделей всех объектов сцены;
- необходимость постоянно отслеживать взаимное положение объектов в составе сцене, в частности, при создании 3D-анимации.

Преимущества 3D-графики:

высокая информативность отдельных зон экрана;

преимущества при вращении объекта;

новые возможности перспективы;

новые формы диаграмм;

влияние на физические реакции зрителя.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключаются преимущества и недостатки векторной графики, по сравнению с пиксельной графикой?
2. Произойдет ли ухудшение четкости векторного изображения при увеличении его размера?
3. Для чего служит Status Bar (Строка состояния)?
4. Какие варианты действий приводят к выделению нескольких объектов?

5. Какие существуют типы узлов на кривой Безье?
6. Что входит в предметную область компьютерной графики?
7. Для каких целей служат графические редакторы?
8. Что понимается под термином “редактирование изображений”?
9. Как соотносятся предметы компьютерной графики и анимации?
10. Как и в каких областях используется деловая графика?
11. Назовите главные элементы интерфейса КОМПАС-3D.
12. Объясните назначение клавиатурных привязок
13. Что такое «графическое моделирование»?
14. Какие есть виды графического моделирования?
15.Какие задачи выполняет графическое моделирование?
16. Чем отличаются стартовые окна в программе SketchUp?
17. Компоненты графических систем.
18. Способы создания геометрических моделей.
19. Геометрические модели хранения и визуализации.
20. Типы геометрических моделей.
21. Виды простейших геометрических элементов и основные способы их создания. Создание геометрических элементов с использованием отношений (общий и частный способы). Создание геометрических элементов с помощью преобразования.
22.Что подразумевает тип копирования «Instance»
23.Способы моделирования в программном пакете 3DSMax
24. Виды графического моделирования
25. Классификация систем графического моделирования
26. Сферы использования систем графического моделирования
27. Что понимается под термином “редактирование изображений”
28. Как соотносятся предметы компьютерной графики и анимации
29. Для каких целей служат графические редакторы
30. Базовые отличия полного пакета Компас 3D от других комплектаций
31. Классификация систем графического моделирования
32. Сферы использования систем графического моделирования
33. Для каких целей служат графические редакторы
34. Что входит в предметную область компьютерной графики
35. С какими форматами работает SketchUp?
36. Какие инструменты нужны для создания объектов

7. **СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ**

В настоящее время идет стремительное развитие направлений разработки систем имитационного моделирования:

- *AnyLogic* – программного обеспечения для имитационного моделирования сложных систем и процессов, позволяющего поддерживать направление агентного моделирования, дискретно-событийного моделирования и разработки моделей системной динамики (разрабатывается российской компанией (англ. XJ Technologies) «Экс Джей Текнолоджис»);
- *GPSS* (англ. General Purpose Simulation System – общецелевой системы моделирования) – языка объектно-ориентированного программирования, используемого для имитационного моделирования систем массового, различных информационных процессов и разработки имитационных моделей в сети интернет;
- *Arena* – разрабатываемого компанией Systems Modeling Corporation программного обеспечения для имитационного моделирования, позволяющего создавать подвижные компьютерные модели, используя которые можно адекватно представить очень многие реальные системы;
- *Plant Simulation* – программной среды имитационного моделирования систем и процессов, предназначенного для оптимизации материало-потоков, загрузки ресурсов, логистики и метода управления для всех уровней планирования от целого производства и сети производств до отдельных линий и участков;
- *SimBioSys: C++* – оболочки агентно-базового эволюционного моделирования в биологических и общественных науках;
- системы моделирования *SWARM* и его расширения *MAML* (Multi-Agent Modelling Language) для моделирования искусственного мира;
- *Ascape*(Agent Landscape) # RePast (Recursive Porous Agent Simulation Toolkit), написанных на платформе языка Java, для поддержки агентно-базового моделирования;
- *NetLogo* # MIMOSE (Micro – and Multilevel Modelling Software) – систем, предназначенных для создания имитационных моделей и технологий моделирования в общественных науках;
- *SPSS, Statistica, PilGrim, Z-Tree* – систем моделирования для исследования экономических статистических явлений и процессов.

Имитационное моделирование (ситуационное моделирование) – метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

Имитационное моделирование – это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация – это постижение сути явления, не прибегая к экспериментам на реальном объекте).

Имитационное моделирование – это частный случай математического моделирования. Существует класс объектов, для которых по различным причинам не разработаны аналитические модели либо не разработаны методы решения полученной модели. В этом случае аналитическая модель заменяется имитатором или имитационной моделью.

Имитационным моделированием иногда называют получение частных численных решений сформулированной задачи на основе аналитических решений или с помощью численных методов.

Имитационная модель – логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

К имитационному моделированию прибегают, когда:

- дорого или невозможно экспериментировать на реальном объекте;
- невозможно построить аналитическую модель: в системе есть время, причинные связи, последствие, нелинейности, стохастические (случайные) переменные;
- необходимо съимитировать поведение системы во времени.

Цель имитационного моделирования состоит в воспроизведении поведения исследуемой системы на основе результатов анализа наиболее существенных взаимосвязей между её элементами или другими словами – в разработке симулятора (англ. *simulation modeling*) исследуемой предметной области для проведения различных экспериментов.

7.1. **Виды имитационного моделирования**

Наиболее распространено агентное моделирование – относительно новое направление в имитационном моделировании (получило распространение в 1990е-2000-е гг.), которое используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами (как в других парадигмах моделирования), а наоборот, когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы. Цель агентных моделей – получить представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном, частном поведении её отдельных активных объектов и взаимодействии этих объектов в системе. Агент – некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться.

Дискретно-событийное моделирование – подход к моделированию, предлагающий абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы, такие как: «ожидание», «обработка заказа», «движение с грузом», «разгрузка» и другие. Дискретно-событийное моделирование наиболее развито и имеет огромную сферу приложений – от логистики и систем массового обслуживания до транспортных и производственных систем. Этот вид моделирования наиболее подходит для моделирования производственных процессов. Основан Джеффри Гордоном в 1960-х годах.

Системная динамика – парадигма моделирования, где для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная на основе этих диаграмм модель имитируется на компьютере. По сути, такой вид моделирования более всех других парадигм помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями. С помощью системной динамики строят модели бизнес-процессов, развития города, модели производства, динамики популяции, экологии и развития эпидемии. Метод основан Джеем Форрестером в 1950 –х годах.

7.2. **Конструирование объектов с ипользованием пакета** GPSS World

GPSS World – общецелевая система имитационного моделирования.

Система GPSS World, разработанная компанией Minuteman Software (США) – это мощная среда компьютерного моделирования общего назначения, разработанная для профессионалов в области моделирования. Это комплексный моделирующий инструмент, охватывающий области как дискретного, так и непрерывного компьютерного моделирования, обладающий высочайшим уровнем интерактивности и визуального представления информации.

Использование *GPSS World* дает возможность оценить эффект конструкторских решений в чрезвычайно сложных системах реального мира.

GPSS World основан на оригинальном языке компьютерного моделирования GPSS, что означает General Purpose Simulation System – общецелевая система моделирования. В основном этот язык был разработан Джефри Гордоном приблизительно в 1960 году в IBM и привнес множество важных концепций в каждую из коммерческих реализаций языков компьютерного моделирования дискретных событий, разработанных с тех пор. *GPSS World* – это прямое развитие языка моделирования GPSS/PC, одной из первых реализаций GPSS для персональных компьютеров. После своего появления в 1984 году GPSS/PC и его последующие версии сохранили тысячам пользователей миллионы долларов. В настоящее время версия *GPSS World* для ОС Windows имеет расширенные возможности, включая пользовательскую среду с интегрированными функциями работы с Интернет.

GPSS World разработан для оперативного получения достоверных результатов с наименьшими усилиями. В соответствии с этими целями в *GPSS World* хорошо проработана визуализация процесса моделирования, а также встроены элементы статистической обработки данных. Сильная сторона *GPSS World* – это его прозрачность для пользователя.

Прозрачность для пользователя ценна по трем причинам. Во-первых, опасно полагаться на непрозрачное моделирование типа «черный ящик», внутренние механизмы функционирования которого скрыты от пользователя. Мало того, что в этом случае нельзя быть уверенным, подходит ли оно для какого-либо конкретного случая, но и невозможно гарантировать, что оно работает, как задумано. Во-вторых, удачные имитационные модели являются очень ценными и пригодны в течение длительного периода времени. Возможно, потребуется, чтобы новые сотрудники ознакомились с внутренними процессами модели, а это почти невозможная задача, если модель не имеет высокого уровня прозрачности. В-третьих, одним из наиболее эффективных, но наименее известных преимуществ компьютерного имитационного моделирования является возможность проникновения в самую суть поведения системы, когда опытный профессионал в области моделирования может видеть внутреннюю динамику в наиболее важные моменты времени процесса моделирования.

GPSS World был разработан с целью решить все эти проблемы. *GPSS World* является объектно-ориентированным языком. Его возможности визуального представления информации позволяют наблюдать и фиксировать внутренние механизмы функционирования моделей. Его интерактивность позволяет одновременно исследовать и управлять процессами моделирования. С помощью встроенных средств анализа данных можно легко вычислить доверительные интервалы и провести дисперсионный анализ. Кроме того, теперь есть возможность автоматически создавать и выполнять сложные отсеивающие и оптимизирующие эксперименты.

GPSS World был разработан, чтобы полностью использовать возможности вычислительной системы. Использование механизма виртуальной памяти позволяет моделям реально достигать размера миллиарда байт. Вытесняющая многозадачность и многопоточность обеспечивают высокую скорость реакции на управляющие воздействия и дают возможность *GPSS World* одновременно выполнять множество задач. Это также означает, что система моделирования *GPSS World* может использовать вычислительные возможности, предоставляемые симметричными многопроцессорными архитектурами (SMP).

GPSS World сочетает в себе функции дискретного и непрерывного моделирования. Возможность перехода из дискретной фазы моделирования в непрерывную фазу и обратно обеспечивает тесную связь с непрерывным моделированием. В непрерывной фазе могут быть установлены пороговые значения, управляющие созданием транзактов в дискретной фазе.

Последняя версия *GPSS World* включает в себя массу нововведений, позволяющих проводить более эффективные исследования и сделать работу с системой максимально простой и удобной для пользователя.

7.3. **Конструирование объектов с ипользованием пакета** Arena

Arena — система имитационного моделирования, которая позволяет создавать динамические модели разнородных процессов и систем, оптимизировать построенную модель. Программа Arena снабжена удобным объектно-ориентированным интерфейсом, обладает широкими функциональными возможностями по адаптации к различным предметным областям.

Основой технологии моделирования *Arena* являются язык моделирования SIMAN и анимационная система Cinema Animation. Отличается гибкими и выразительными средствами моделирования. Отображение результатов моделирования в *Arena* выполняется с использованием Cinema Animation. Процесс моделирования организован следующим образом. Сначала пользователь шаг за шагом строит в визуальном редакторе программы *Arena* модель. Затем система генерирует по ней соответствующий код на SIMAN, после чего автоматически запускается Cinema Animation.

Arena состоит из блоков моделирования (модули) и операций (сущности). Сущности двигаются между модулями по мере их обслуживания.

Эффект от создания имитационных моделей увеличивается благодаря предварительному анализу бизнес-процессов. Таким образом, функциональные и имитационные модели дополняют друг друга, при этом они могут быть тесно взаимосвязаны. Имитационная модель дает больше информации для анализа системы, в свою очередь результаты такого анализа могут стать причиной модификации модели процессов. Наиболее целесообразно сначала создать функциональную модель, а затем на ее основе строить модель имитационную. Для поддержки такой технологии инструментальное средство функционального моделирования BPwin имеет возможность экспорта диаграммы IDEF3 в имитационную модель *Arena*.

*7.3.1. **Построение простой имитационной модели***

Построим простую имитационную модель в программе *Arena* на примере рабочей станции (рис. 7.1). Время поступления запросов в систему экспоненциально распределено, в случае занятости обслуживающего устройства запрос встает в очередь. Время обслуживания экспоненциально распределено со средним значением в 24 минуты.

Переместим модули Create, Process и Dispose в окно рабочего модуля.

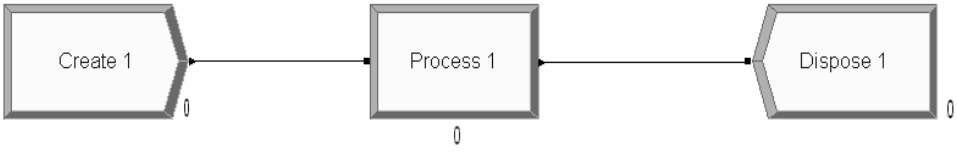


Рисунок 7.1 - Имитационная модель работы рабочей станции

Для задания свойств графическому модулю дважды щелкнем по нему и в диалоге (рис. 7.2) зададим значения параметров в соответствии с условием.

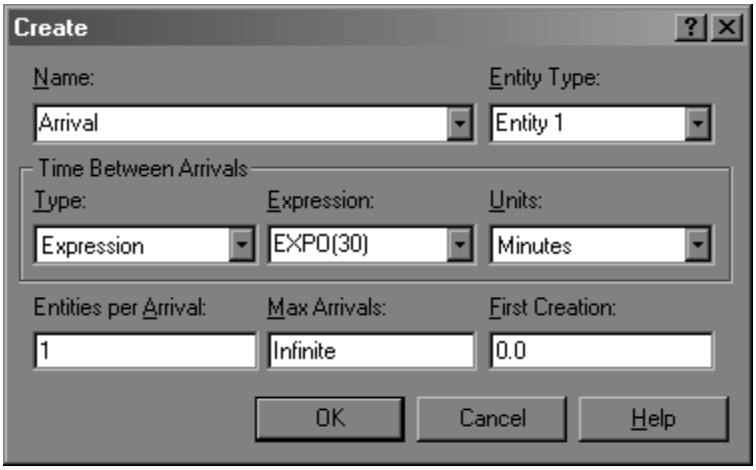


Рисунок 7.2 — Диалоговое окно свойств модуля Create

Поле Resources определяет ресурсы или группы ресурсов, которые будут обрабатывать сущности в этом модуле. Добавление ресурса осуществляется с помощью кнопки Add, в появившемся окне укажем использование одного ресурса (рис. 7.3, 7.4, 7.5).

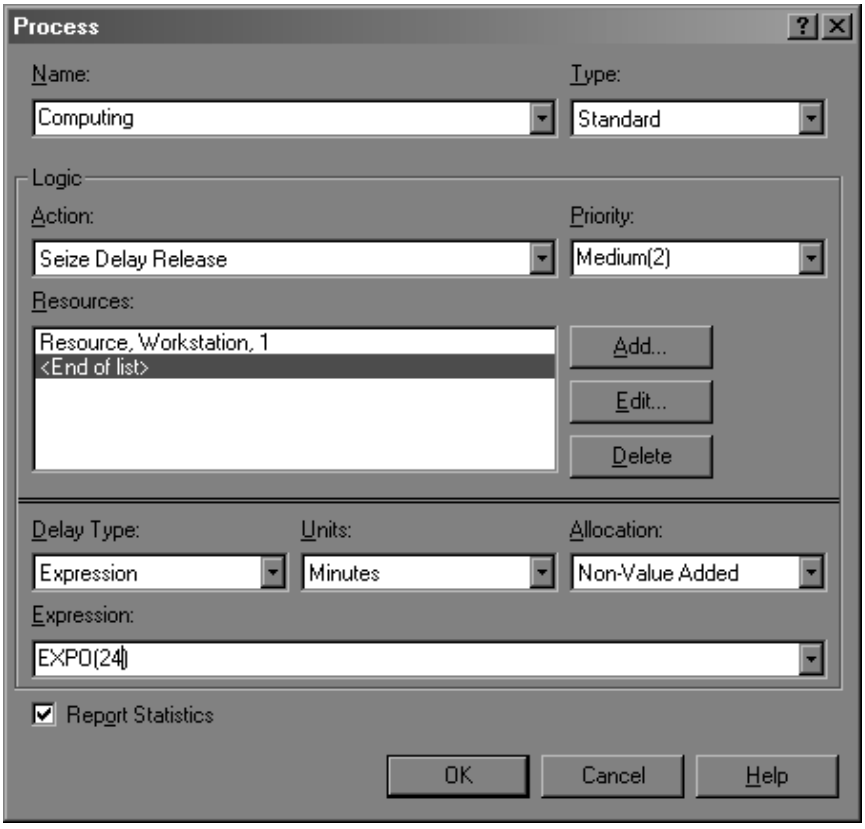


Рисунок 7.3 — Диалоговое окно свойств модуля Process

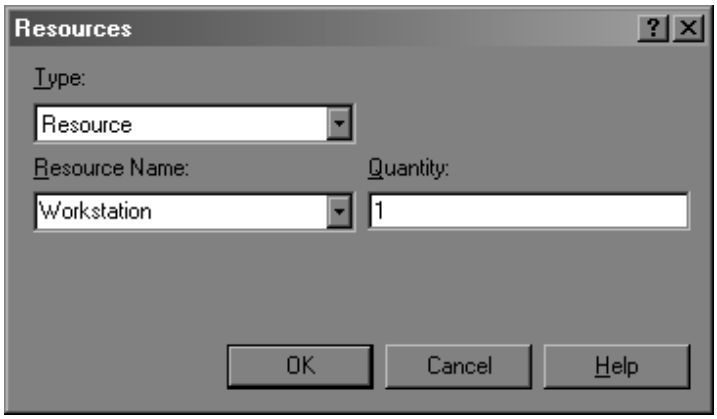


Рисунок 7.4 — Диалоговое окно задания ресурсов в модуле Process

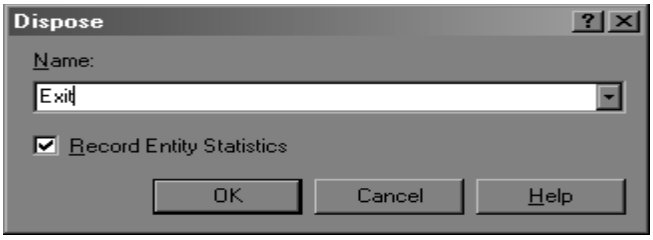


Рисунок 7.5 — Диалоговое окно свойств модуля Dispose

После задания каждого модуля модель принимает вид (рис. 7.6):

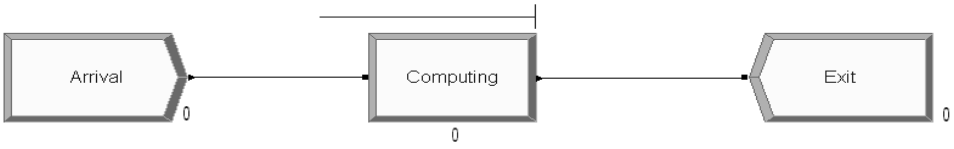


Рисунок 7.6 — Имитационная модель работы рабочей станции

Для задания длительности моделирования перейдем в меню Run/Setup (рис.7.7). В поле ReplicationLength установим длительность 5000, а в поле TimeUnits единицу измерения времени Minutes. В BaseTimeUnits также указываем Minutes для генерации отсчета в минутах.

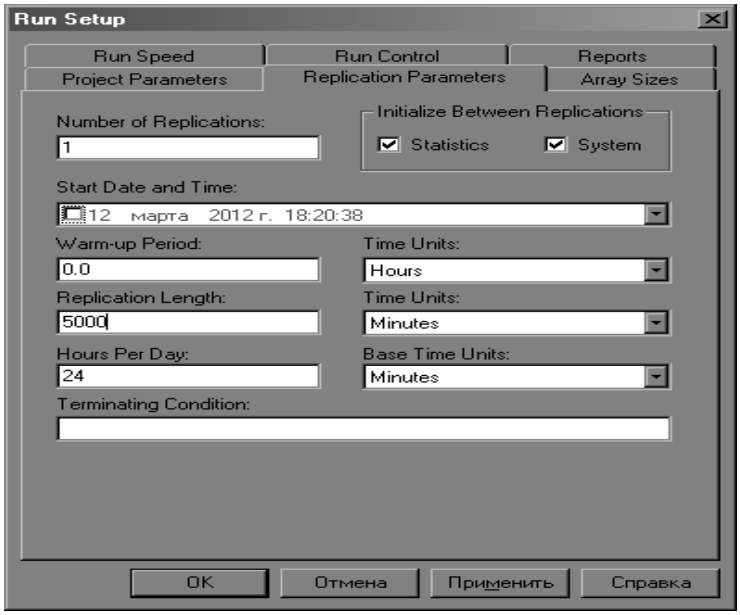


Рисунок 7.7 — Окно параметров моделирования

Проигрывание модели начнем командой Run/Go (рис. 7.8).

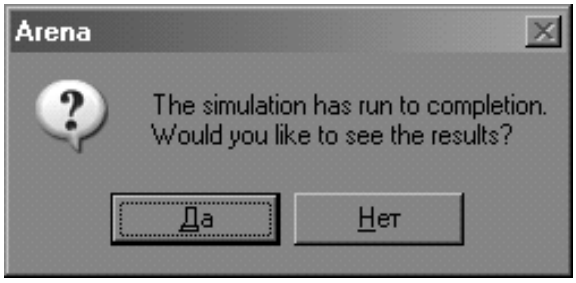


Рисунок 7.8 — Окно, появляющееся по завершению моделирования

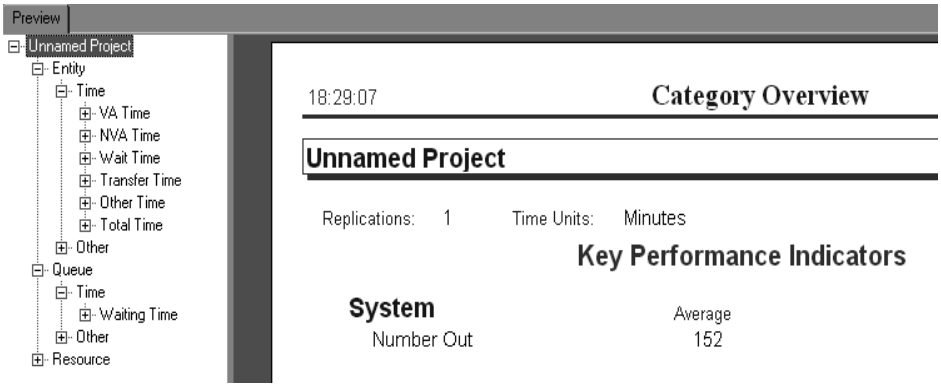


Рисунок 7.9 — Отчет по результатам проигрывания модели, дерево параметров

Результаты моделирования модели сведены в табл. 7.1

Таблица 7.1. Результаты моделирования модели

Характеристика	Где найти	Значение
Средняя продолжительность пребывания запросов в системе	**Панель слева** — Preview Entity — Time — Total Time (Average) Total Time Average Entity 1 127.30 — 127,30 мин	
Среднее число запросов в очереди	Queue — Other — Number Waiting (Average) Number Waiting Average Computing.Queue 3.0727 — 3,07 запросов	
Среднее число запросов на обработке	Resourse — Usage — Number Busy (Average) Number Busy Average Workstation 0.7977 — 0,80 запросов	
Среднее число запросов в системе	**Среднее число** запросов в очереди (NumberWaiting) + среднее число запросов на обработке (NumberBusy)	3,07+0,80=3,87

По результатам моделирования видно, что СМО работает стационарно, т.е. не образуется бесконечной очереди; среднее число запросов в системе, равное 3,87, можно считать удовлетворительным.

Для повторного проигрывания модели необходимо остановить предыдущую симуляцию командой Run/End.

7.3.2. Моделирование работы системы обслуживания покупателей на кассе супермаркета

Смоделируем работу системы обслуживания покупателей на кассе супермаркета, если известно, что поток покупателей имеет пуассоновское распределение со средним значением 5 мин (обозначается POIS(5)), а время обслуживания на кассе занимает от 2 до 10 мин с наиболее вероятным значением 3 мин (используется распределение Triangular). Какое среднее время ожидания покупателей в очереди, если длительность моделирования составляет 15 ч?

Переместим модули Create, Process и Dispose в окно рабочего модуля (рис. 7.10). Для задания свойств каждого графического модуля дважды щелкнем по нему и в диалоге зададим значения параметров в соответствии с условием (рис. 7.10, 7.11, 7.12).

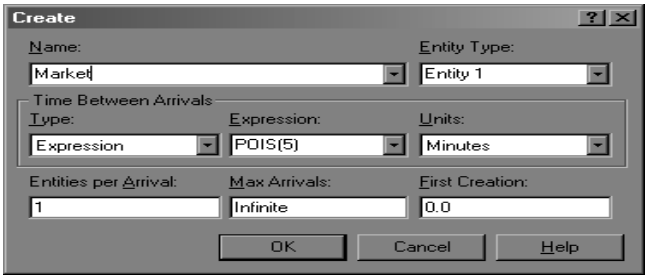


Рисунок 7.10 — Диалоговое окно свойств модуля Create

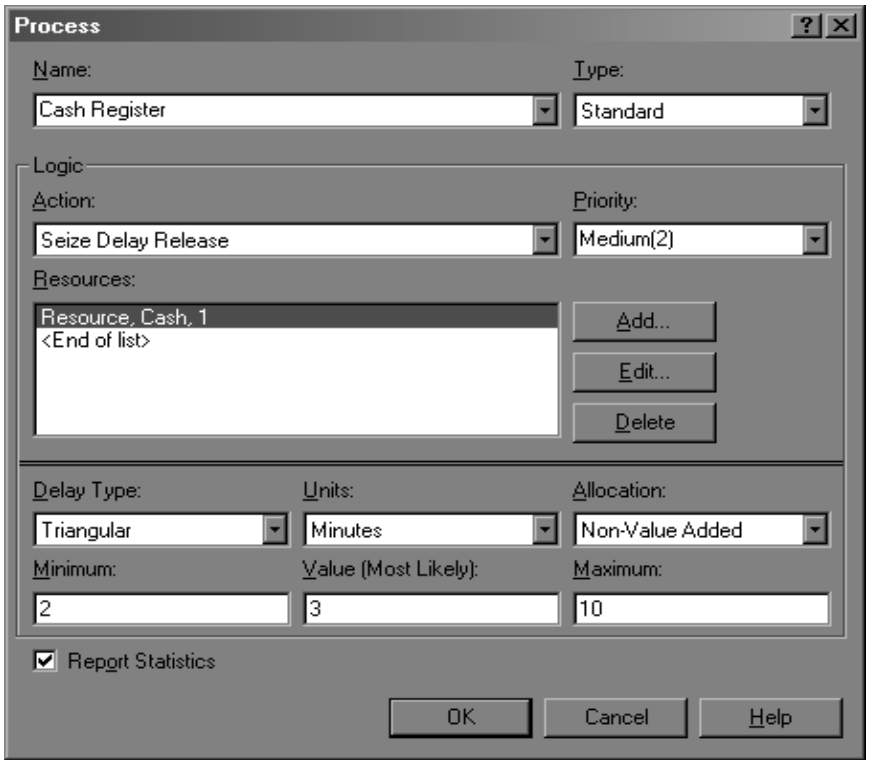


Рисунок 7.11 — Диалоговое окно свойств модуля Process

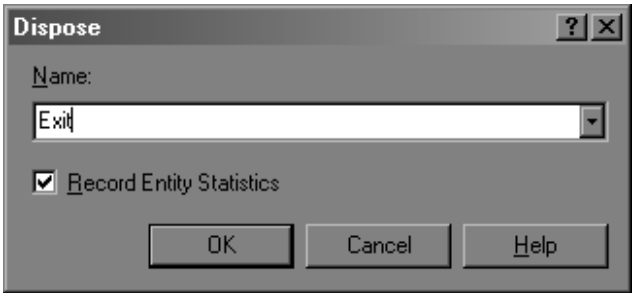


Рисунок 7.12 — Диалоговое окно свойств модуля Dispose

После задания каждого модуля модель принимает вид (рис. 7.13):

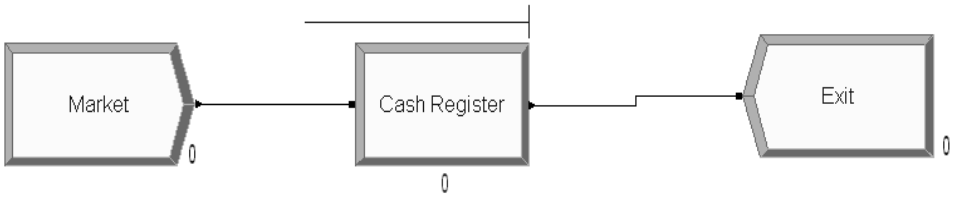


Рисунок 7.13 — Имитационная модель системы обслуживания покупателей на кассе супермаркета

Для задания длительности моделирования перейдем в меню Run/Setup (рис. 7.14). В поле ReplicationLength установим длительность 900, а в поле TimeUnits единицу измерения времени Minutes. В BaseTimeUnits также указываем Minutes для генерации отчета в минутах.

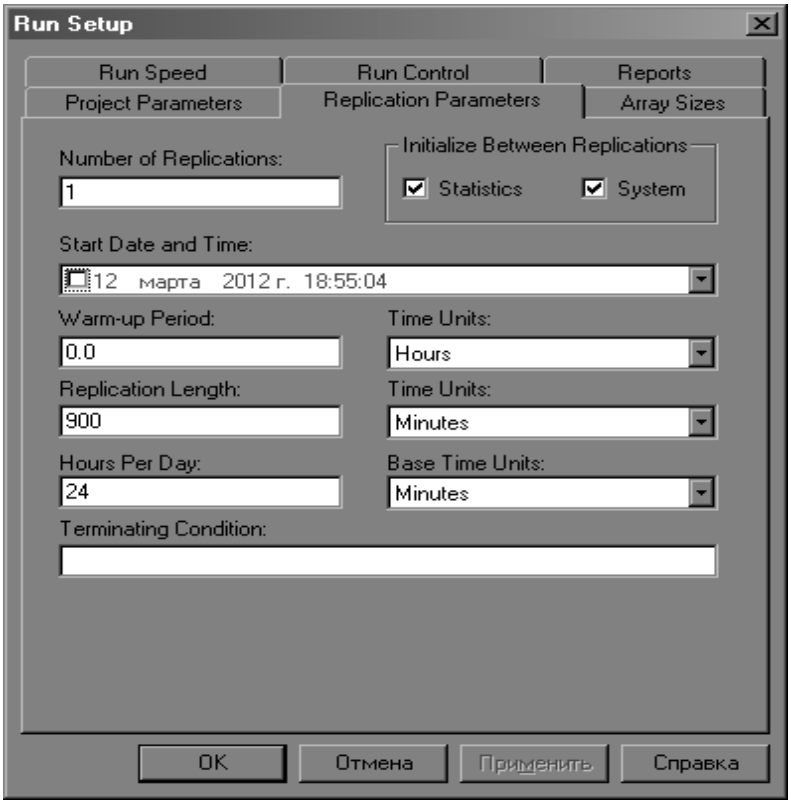


Рисунок 7.14 — Окно параметров моделирования

Результаты моделирования модели сведены в табл. 7.2

Таблица 7.2. Результаты моделирования модели

Характеристика	Где найти	Значение
Средняя продолжительность ожидания покупателей в очереди	Queue — Time — Waiting Time (Average) Waiting Time Average Cash Register.Queue 16.9295	16,9 минут

Модель (рис. 7.15) показывает систему обработки документа (закладной). Сначала документ регистрирует секретарь (иконка слева в нижней части рисунка, затем просматривает клерк (иконка справа). Затем клерк либо принимает документ, либо возвращает.

Очередь документов показывается в виде набора иконок сверху от процесса Review Application и в виде графика в правой нижней части рисунка.

85,7 % заявлений принят, 13,6 % заявлений отклонен.

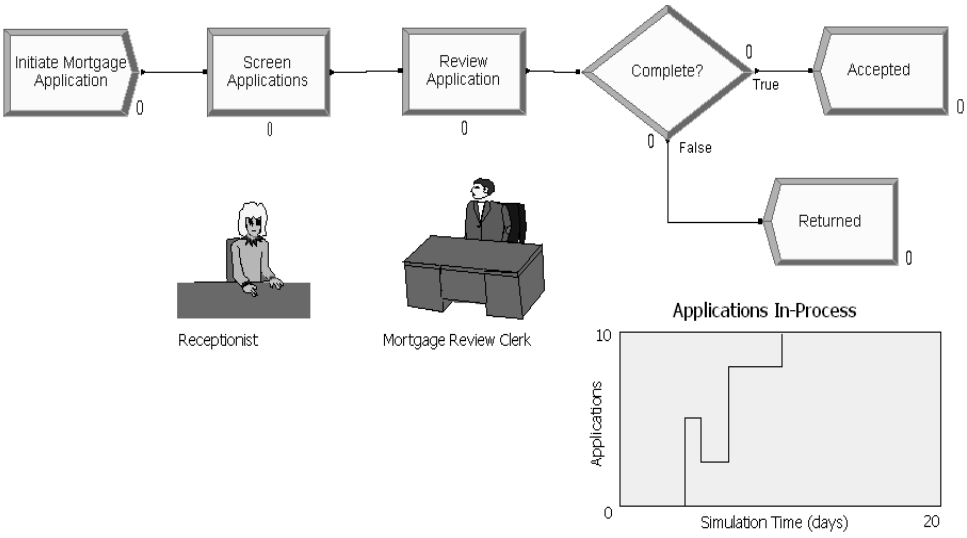


Рисунок 7.15 — Модель обработки документа «Mortgage Extention 1»

*7.3.3. **Построение модели** IDEF3*

Запустим программу BPwin, в появившемся окне укажем имя файла и тип диаграммы Process Flow (IDEF3) (рис. 7.16).

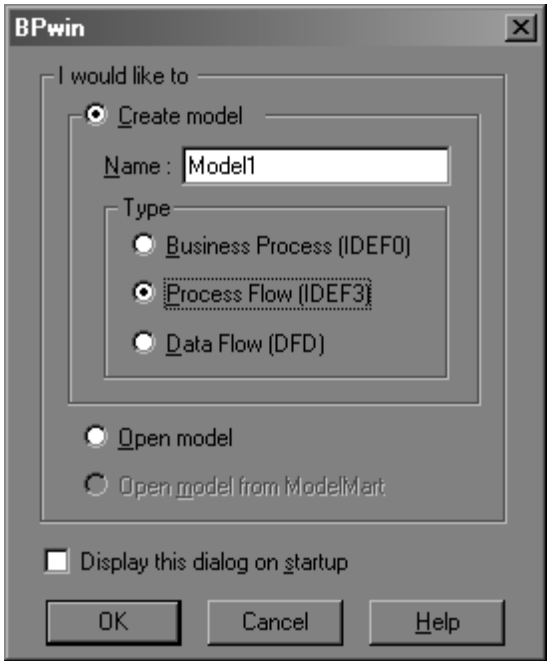


Рисунок 7.16 — Диалоговое окно при создании новой диаграммы

Рассмотрим пример построения IDEF3 модели «Диагностика автомобилей» (рис. 7.17) для дальнейшего экспорта в *Arena*. При построении процессной модели используется ряд особенностей. Для задания начальных и конечных блоков процессной модели используется Referent tool.

Названия блоков указываются на английском языке или транслитом, так как *Arena* не распознает кириллицу.

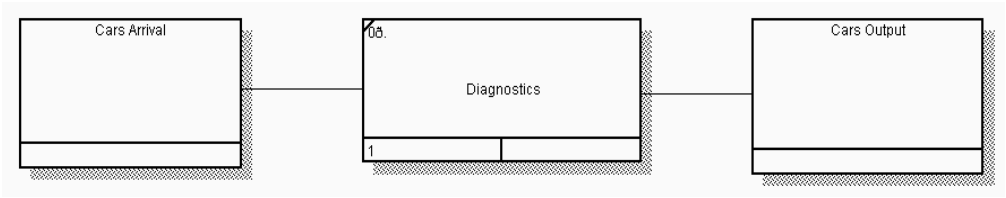


Рисунок 7.17 — Процессная модель «Диагностика автомобилей»

Стрелки от начальных к конечным блокам задаются в стиле Referent. Свойства стрелки (Arrow Properties) в диалоговом окне смотреть на рис. 7.18.

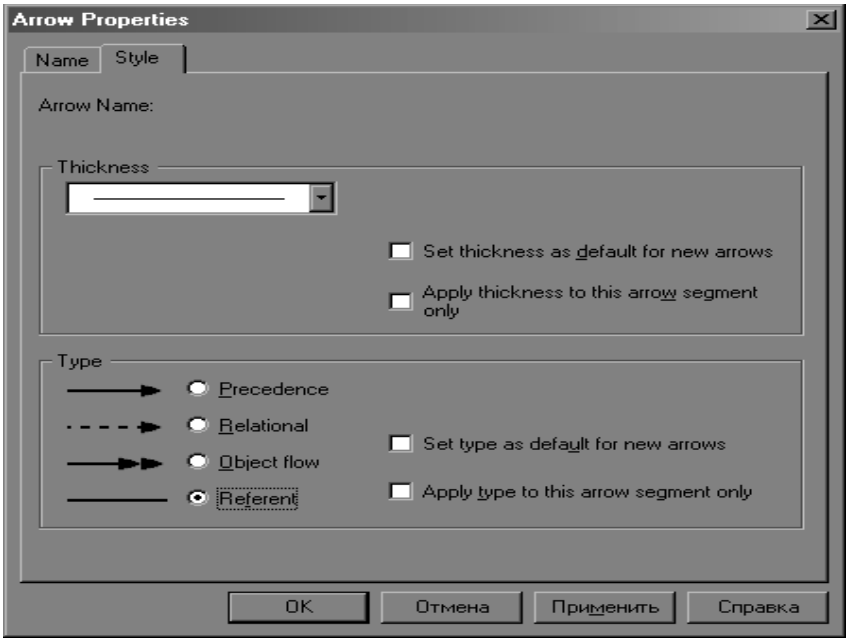


Рисунок 7.18 — Свойства стрелки (Arrow Properties)

Поскольку имитационная модель *Arena* должна содержать дополнительные параметры по сравнению с моделью IDEF3, в BPwin используются свойства User-Defined Properties (UDP), импорт которых предварительно осуществляется из файла ArenaBEUDPs.bp1.

Для этого необходимо открыть модель ProgramFiles / ComputerAssociates / BPwin / Samples / Arena / ArenaBEUDPs.bp1 и, находясь в только что созданной модели с примером «Диагностики автомобилей», импортировать настройки командой Model/MergeModelDictionaries/ (рис. 7.19).

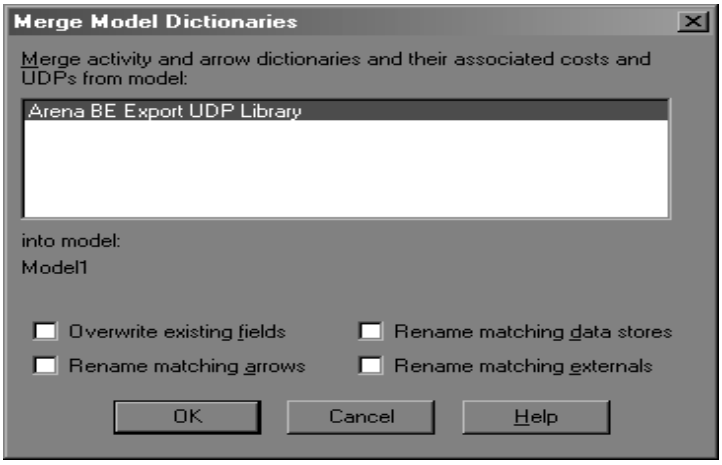


Рисунок 7.19 – Диалоговое окно Merge Model Properties

В результате в новой модели появятся UDP настройки (Dictionary/UDP/).

Устанавливаем UDP настройки для каждого блока. Блок Cars Arrival (рис. 7.20), диалоговое окно которого показано ниже, в динамической модели будет использоваться для генерирования приезда автомобилей на диагностику. Укажем, что интервалы времени между поступлением деталей имеют пуассоновское распределение со средним значением 1 час, что обозначается как POIS (1).

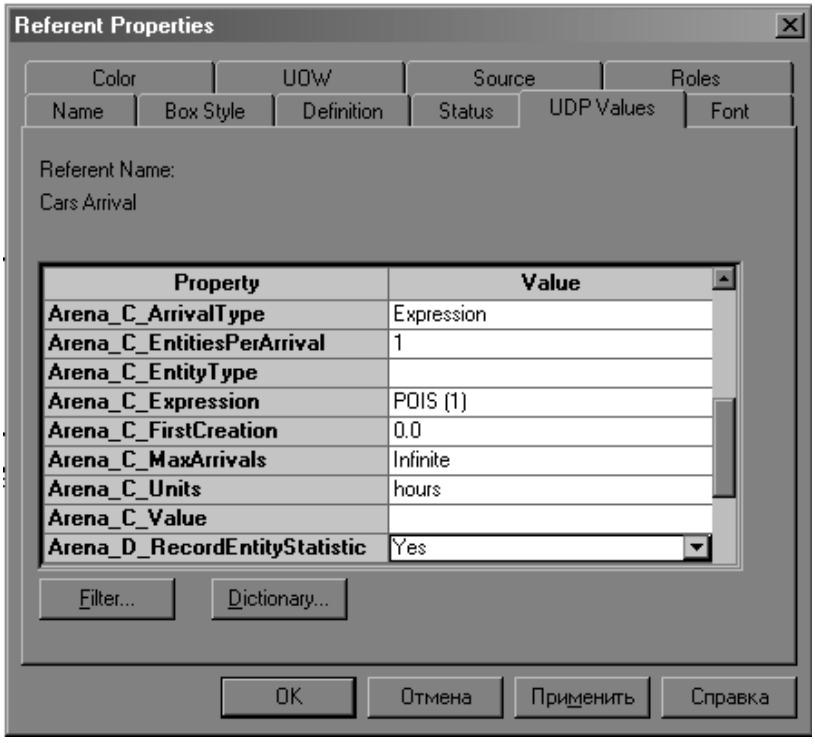


Рисунок 7.20 — UDP свойства блока Cars Arrival

Блок Cars Arrival соединяется с блоком Diagnostics (рис. 7.21), в котором происходит процесс диагностики автомобилей. Продолжительность диагностики экспоненциально распределена со средним значением 0,7 ч.

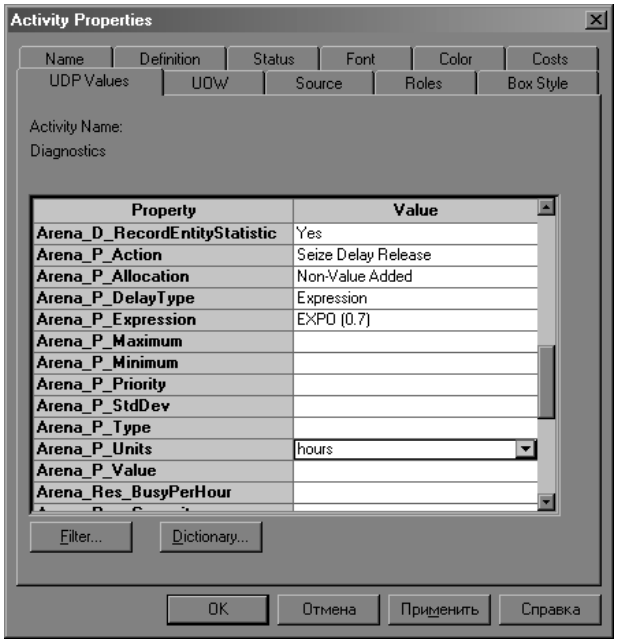


Рисунок 7.21 — UDP свойства блока Diagnostics

В конечном блоке Cars Output (рис. 7.22) указывается только галочка о сборе статистики.

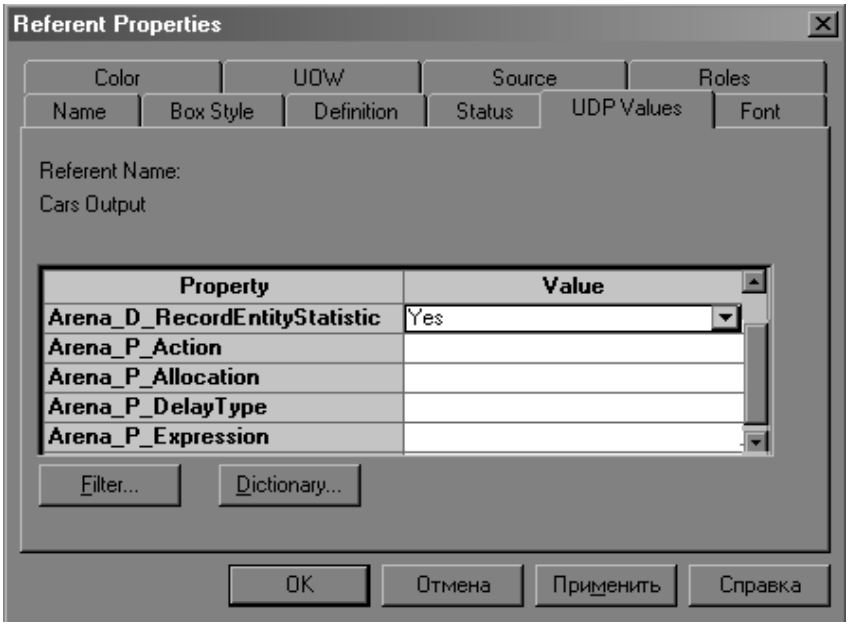


Рисунок 7.22 — UDP свойства блока Cars Output

После указания UDP на каждом блоке появляется скрепка (рис. 7.23):

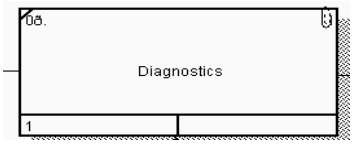


Рисунок 7.23 — Блоки с установленными UDP

Для успешного проигрывания модели необходимо добавить ресурс (люди, оборудование), который проводит диагностику. Ресурс задается при помощи стрелки «механизм», присоединенной к нижней стороне блока работы (рис.7.24). Стрелка имеет стиль Relational.

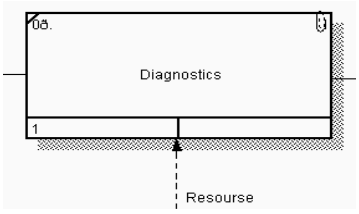


Рисунок 7.24 — Блок Diagnostics со стрелкой Resource

После задания имени стрелки появляется возможность указания ее дополнительных свойств. На вкладке UDP Values (рис.7.25) вписывается название ресурса и его количество. В нашем примере: ресурс — один мастер по диагностике.

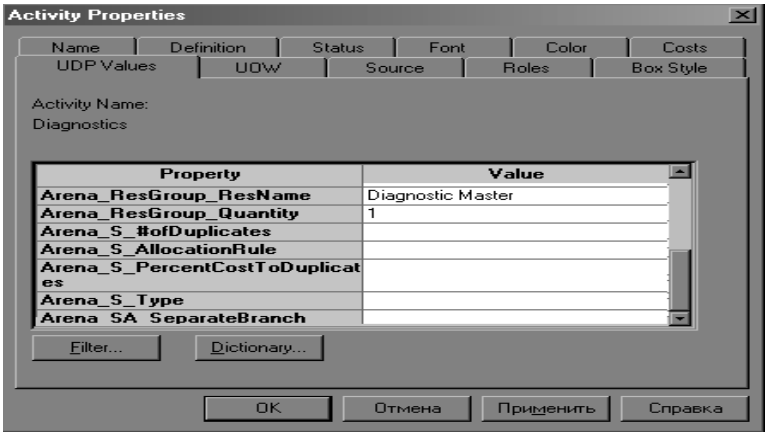


Рисунок 7.25 — Модель «Диагностика автомобилей» в IDEF3

Мастер может проводить диагностику только одного автомобиля в каждый момент времени; если мастер занят, автомобили встают в очередь и ждут, пока он освободится.

Перед экспортом в *Arena* модель в IDEF3 примет вид (рис. 7.26):

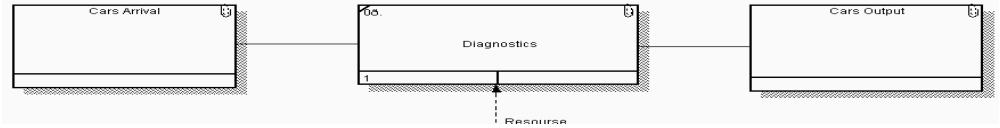


Рисунок 7.26 — Модель «Диагностика автомобилей» в IDEF3

*7.3.4. **Экспорт в** Arena*

Экспорт модели в Arena осуществляется командой File/Export/Arena. При завершении экспорта выводится сообщение:

В результате экспорта получим модель в пакете Arena (рис. 7.27):

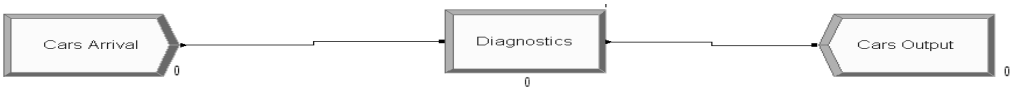


Рисунок 7.27 — Имитационная модель в пакете Arena

Откроем окно параметров моделирования (рис. 7.28) командой Run/Setup. Установим длительность моделирования, равную 100 ч.

Рисунок 7.28 — Окно установки параметров моделирования

Результаты моделирования модели сведены в табл.7.3

Таблица 7.3. Результаты моделирования модели

Характеристика	Где найти	Значение
Средняя продолжительность пребывания запросов в системе	Панельслева — Preview Entity — Time — Total Time (Average) Total Time — Average: 2,96 ч Entity 1 — 2.9578	
Среднее число запросов в очереди	Queue — Other — Number Waiting (Average) Number Waiting — Average: 2,02 машины Diagnostics.Queue — 2.0242	
Средняя продолжительность пребывания запросов в очереди	Queue — Time — Waiting Time (Average) Waiting Time — Average: 2,1 ч Diagnostics.Queue — 2.1085	
Среднее число запросов на обработке	Resourse — Usage — Number Busy (Average) Number Busy — Average: 0,78 машин Diagnostics Master — 0.7858	
Среднее число запросов в системе	Среднее число запросов в очереди + среднее число запросов на обработке	2,02+0,78=2,8 машин

7.4. **Сравнительный анализ результатов имитационного моделирования и аналитического решения**

Представим задачу на диагностику автомобилей в терминах теории СМО. СМО имеет один канал обслуживания (мастер по диагностике). Входящий поток машин на обслуживание — простейший пуассоновский поток с интенсивностью λ=1. Интенсивность потока обслуживания равна μ. Длительность обслуживания — случайная величина, подчиненная показательному закону распределения со средним значением 0,7 ч. Рассчитаем характеристики одноканальной СМО с ожиданием, без ограничения на длину очереди:

$$\mu = \frac{1}{T_{\text{обслуж}}} = \frac{1}{0{,}7} = 1{,}4286$$

$$\psi = \frac{\lambda}{\mu} = \frac{1}{1{,}4286} = 0{,}6999$$

$\psi = 0{,}6999 < 1$, т.е. условие стационарности СМО выполняется.

Среднее число машин в системе:

$$L_S = \frac{\psi}{1-\psi} = \frac{0{,}6999}{1-0{,}6999} = \frac{0{,}6999}{0{,}3001} = 2{,}33$$

Средняя продолжительность пребывания машин в системе:

$$W_S = \frac{1}{\mu \cdot (1-\psi)} = \frac{1}{1{,}4286 \cdot (1-0{,}6999)} = \frac{1}{0{,}4287} = 2{,}33$$

Среднее число машин в очереди:

$$L_q = \frac{\psi^2}{1-\psi} = \frac{(0{,}6999)^2}{1-0{,}6999} = \frac{0{,}4899}{0{,}3001} = 1{,}63$$

Средняя продолжительность пребывания машин в очереди:

$$W_q = \frac{\psi}{\mu \cdot (1-\psi)} = \frac{0{,}6999}{1{,}4286 \cdot (1-0{,}6999)} = \frac{0{,}6999}{0{,}4287} = 1{,}63$$

Сравним полученный результаты аналитического решения с результатами имитационного моделирования (табл.7.4).

Таблица 7.4. — Сравнительный анализ

Показатели	Результаты имитационного моделирования				Результаты аналитического решения
	100 ч	300 ч	1000 ч	1500 ч	
1. Среднее число машин на обслуживание в системе	2,8	2,76	2,4	2,3	$L_s = 2.33$
2. Средняя продолжительность пребывания машин в очереди	2,96	2,7	2,4	2,31	$W_s = 2.33$
3. Среднее число машин в очереди на обслуживании	2,02	2,03	1,76	1,66	$L_q = 1.63$
4. Средняя продолжительность пребывания машин в очереди	2,1	1,98	1,71	1,63	$W_q = 1.63$

Как видно из табл. 7.4, результаты имитационного моделирования приближаются к результатам аналитического решения по мере увеличения длительности моделирования.

7.5. **Конструирование объектов с ипользованием пакета**NetLogo

В начале 1990-х годов М. Резник предложил использовать многоагентное сообщество черепашек для освоения учениками экологических стратегий. Со множеством черепашек в языке StarLogo ученики могли наблюдать, изучать и моделировать сложные физические, химические, биологические и социальные феномены. Хотя язык создавался, в первую очередь, как средство обучения, в этой среде оказалось возможным ставить и серьезные эксперименты по многоагентному моделированию. Исследовательские возможности среды получили дальнейшее развитие в языке *NetLogo*. Язык был создан Ури Виленским в 1999 году и продолжает активно развиваться в настоящее время. Среда программирования NetLogo служит для моделирования ситуаций и феноменов, происходящих в природе и обществе. NetLogo удобно использовать для моделирования сложных, развивающихся во времени систем. Создатель модели может давать указания сотням и тысячам независимых «агентов» действующим параллельно. Это открывает возможность для объяснения и понимания связей между поведением отдельных индивидуумов и явлениями, которые происходят на макроуровне.

Язык NetLogo достаточно прост, и ученики, и учителя могут создавать в этой среде свои собственные авторские модели. В то же время это достаточно мощный язык и среда для проведения исследований наряду с такими средствами как Swarm, Repast, MASON. Благодаря мощным вычислительным средствам и относительной простоте синтаксиса *NetLogo*, на его основе в последние годы было построено множество исследовательских моделей, которые использовались и обсуждались в книгах по многоагентному моделированию и моделированию в социологии.

Важной особенностью четвертой версии языка *NetLogo* является появление нового типа агентов. К черепашкам (turtles) и пятнышкам (patches) добавились связи (links). Агенты нового типа открывают новые возможности для моделирования сетевых отношений. Связь в *NetLogo* – это агент, связывающий две черепашки или два узла. Связь создается командой, обращенной к черепашке. Например:

ask turtle 1 [create-link-with turtle 0]

Связи в *NetLogo* бывают двух типов – направленные и ненаправленные. Ненаправленные связи создаются командой create-link-with

Направленные связи создаются командами:

create-link-from

create-link-to

Например, рис. 7.29 создан следующей строкой команд:
ca cro 10 [set size 3 set shape «circle» fd 8] ask turtles [create-links-to other turtles]

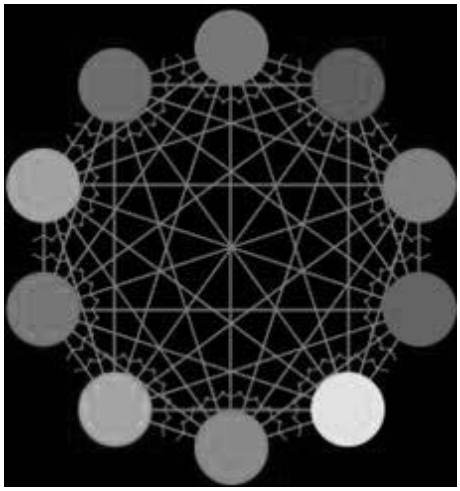


Рисунок 7.29. — Образование направленных связей в среде NetLogo

*7.5.1. **Моделирование организационных отношений с использованием «связей»** Netlogo*

Появление нового типа агентов позволяет рассматривать новые феномены и создавать новые модели, в которых большое значение играют связи между узлами сети. В качестве примера многоагентной сетевой модели, созданной с использованием связей NetLogo, мы построили модель организационных отношений «Лидерство». Исходной посылкой данной модели является положение известного немецкого социолога М. Вебера (1864 – 1920) о существовании качеств, благодаря которым одни люди приобретают способность управлять другими людьми. «Харизмой» следует называть качество личности, признаваемое необычайным, благодаря которому она оценивается как одаренная сверхъестественными, сверхчеловеческими или, по меньшей мере, специфически особыми силами и свойствами, не доступными другим людям. Несмотря на то, что концепция харизмы носит несколько упрощенный характер, она с успехом может быть использована при моделировании управленческих и организационных процессов. В рамках данной модели под харизмой мы будем понимать способность лидера подчинять себе не принадлежащих ни к какой организации агентов, а также перетягивать в свою организацию агентов других организаций.

Модель позволяет выстраивать и анализировать разнообразные сценарии взаимодействия организаций с различными особенностями — стилями лидерства, характерами подчинений и связей, стратегиями и культурами. Пользовательский интерфейс работы с моделью представлен на рис. 7.30.

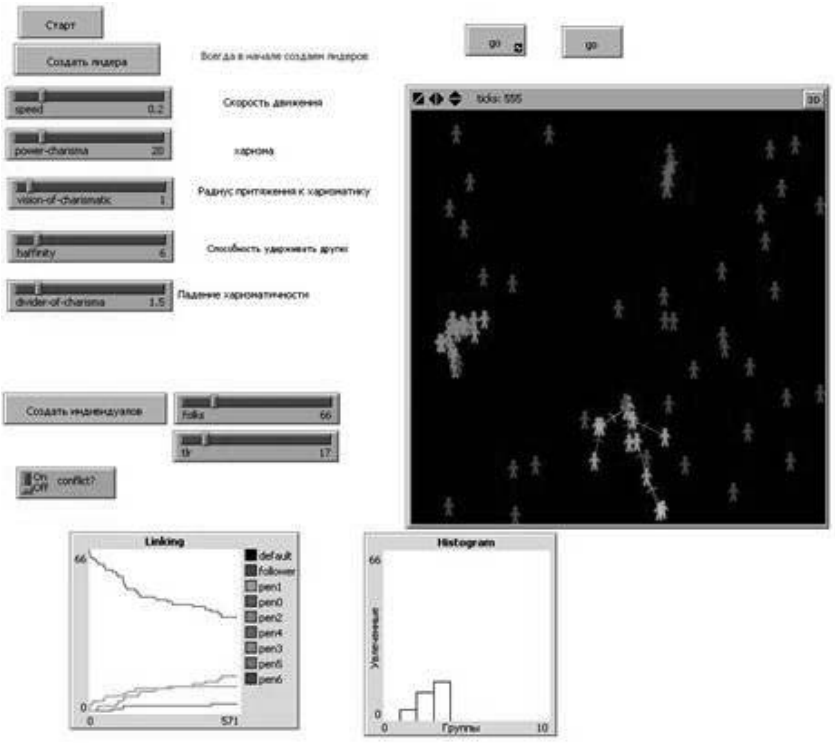


Рисунок 7. 30. — Пользовательский интерфейс модели «Лидерство»

Как устроена модель?

При создании каждый лидер-харизматик получает следующие свойства:

- vision – радиус зоны влияния, в которой он действует на других агентов;
- phil – обозначает номер харизматика. Вначале создаются лидеры. Каждый лидер получает свой номер. Когда лидер привлекает к себе нового участника, то он передает ему этот номер;
- charisma – харизматичность как способность привлекать на свою сторону. Попавший под влияние лидера агент образует с лидером направленную связь (create-link-to), начинает двигаться в одном направлении с лидером и перенимает часть его харизматичности:
- set charisma ([charisma] of myself) / ([divider] of myself);
- divider – степень падения харизматичности у новых членов клана;
- affinity – сколько новых адептов может зацепить харизматик – свойство идеи;
- hcolor – цвет клана;
- ch-speed – скорость перемещения.

Одиночки при создании получают только свойство tolerance – устойчивость к воздействию харизматиков. Значение tolerance варьируется в популяции одиночек в интервале от 0 до значения рычажка tlr.

Харизматичный лидер в зоне своего влияния воздействует на агентов одиночек. Если харизма лидера выше устойчивости одиночки, то одиночка присоединяется к лидеру и образует с ним связь:

[ask min-one-of new_group [distance myself][create-link-to myself set breed charismatics set my_ boss myself]]

Если в системе допустима борьба за уже присоединившихся к лидерам агентов, то лидеры оказывают влияние не только на одиночек, но и на членов других групп. Переход членов групп в другую группу под воздействием другого лидера описывается следующим правилом: Если сила воздействия чужого лидера в два раза превосходит силу воздействия лидера, который привлек меня в группу, то я перехожу в другую группу.

if ([charisma] of my_boss) < ([charisma] of myself) / 2

В начале кнопка Старт (Setup) очищает экран, удаляет всех агентов (turtles, links, patches).

Пользователь выбирает свойства лидера и создает одного или нескольких лидеров со следующими свойствами, которые можно регулировать при помощи рычажков:

- рычажок – speed – скорость передвижения агента;
- рычажок – power-charisma – позволяет подбирать степень харизматичности лидера. Значение charisma сравнивается с толерантностью одиночки (tolerance). Если харизма больше толерантности, то одиночка присоединяется к лидеру. При этом одиночка становится членом команды и наследует некоторые свойства лидера, такие как направление движения и phil – номер идеи или клана;
- рычажок – vision-of-harismatic – расстояние, на котором действует сила харизматика. Все кто в этой зоне, могут попасть под его влияние и стать носителем его идеи – поворачиваются в его направлении и следуют за ним;
- рычажок – haffinity - сколько может удерживать харизмати вокругсебя;
- рычажок – divider-of-harisma — позволяет управлять степенью снижения харизматичности от лидера к следующим членам группы. Каждый следующий носитель идеи не столь же харизматичен, как лидер клана.
- Кнопка «Создать лидера» – create-harismen – создает лидера.
- Кнопка «Создать индивидуалов» – create-flw – создает агентов без исходной харизмы и цели. Число этих агентов регулируется рычажком folks.
- Рычажок tlr позволяет регулировать уровень исходной толерантности агентов-одиночек к воздействию харизматического лидера. Если толерантность выше харизмы, то одиночка не реагирует на лидера. При создании агент получает значение tolerance от 0 до tlr.
- Кнопка «Go» – запускает процесс образования сети/сетей.
- Переключатель Conflict позволяет выбрать два режима модели:
  - off – лидеры не жестко конкурируют между собой. Если одиночка присоединился к лидеру, то он остается в этой группе;
  - on – группы конкурируют между собой. Если харизматичность агента из другого лагеря в два раза выше, чем харизматичность данного агента, то данный агент переходит в другую группу.

График Linking показывает изменение численности групп. Цвет каждой группы агентов совпадает с цветом карандаша:

- Синий – количество одиночек.
- Красный – группа первого лидера phil = 1.
- Зеленый – группа второго лидера phil = 2.
- Оранжевый – группа третьего лидера phil = 3.

Изменение численности агентов, принадлежащих разным группам, представлено на рис. 7.31.

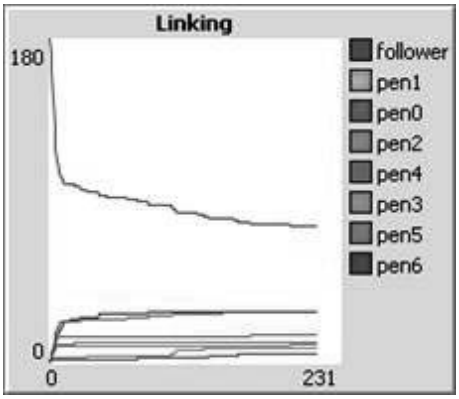


Рисунок 7.31. — Динамика изменения численности групп

*7.5.2. **Эксперименты с различными типами организационных отношений при помощи модели «Лидерство»***

При помощи модели «Лидерство» построим и испытаем различные модели формирования «организации» вокруг харизматического лидера. На рис.7.32 представлена ситуация отсутствия лидера. В этом случае «сотрудники» могут находиться в ситуации броуновского движения как угодно долго, бессмысленно блуждая по полю (ситуация отсутствия стратегии).

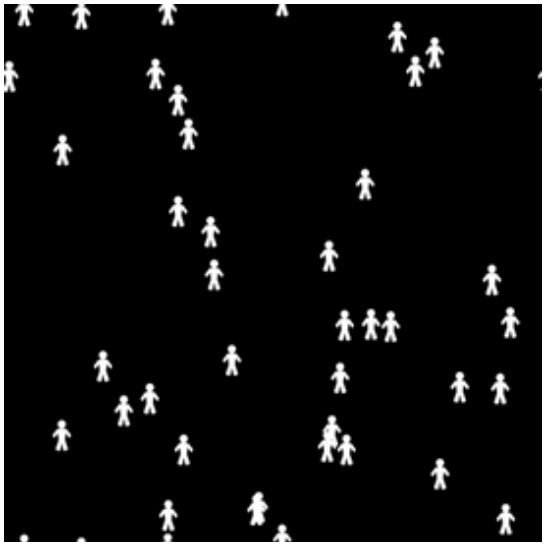


Рисунок 7.32. — Отсутствие лидера

Введем в модель одного лидера. Лидер отличается от других персонажей тем, что он «знает, куда идти» – у него есть некая стратегия движения, которую он предлагает другим агентам. В модели ведущим качеством лидера является «харизма» – его способность увлекать за собой других людей. Если одиночка попадает в поле действия харизмы лидера и харизматичность лидера превышает устойчивость одиночки, то между ними образуется связь и последователь начинает следовать за лидером. Это событие представлено на рис.7.33 и описывается в модели следующим правилом:

```
let new_group followers in-radius my_vision with [tolerance < [charisma] of myself]
if (count new_group) > 0
[ask min-one-of  new_group [distance myself]
[set heading [heading] of myself  create-link-to myself set breed charismatics set my_boss myself]
```

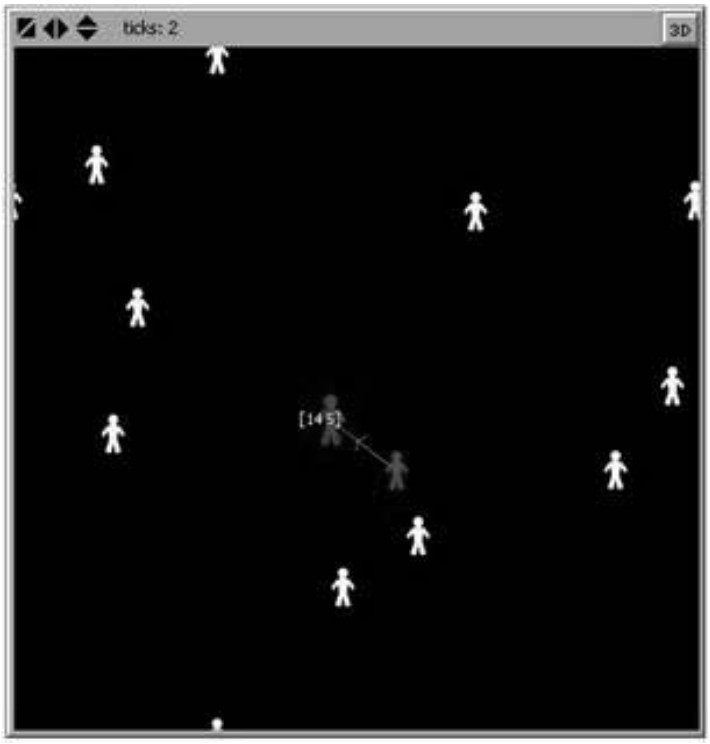


Рисунок 7.33 — Присоединение одиночки к лидеру

Как мы видим, у появившегося последователя меняется цвет – у модели он такой же, как и у лидера. Интенсивность оттенка отображает то, какую степень харизмы передал ему лидер. Если цвет такой же, это значит, что последователю присуща такая же степень харизмы, как и лидеру, если он значительно бледнее – то и степень харизмы ниже. Вместе с цветом последователь наследует и способность присоединять к себе других последователей. При низкой толерантности одиночек через некоторое время в системе несвязанных агентов не остается; все они оказываются объединены в «организацию» с определенной структурой. Характер и форма этой структуры определяются параметрами, заложенными в модели.

Модель «Лидерство» позволяет воспроизвести различные типы организационных отношений и связанные с ними организационные ситуации. Эта популярная типология организационных культур строится на оппозиции двух пар признаков:

1. Гибкость, спонтанность, динамизм vs. Стабильность, порядок, контроль;

2.Внутренний фокус, интеграция, единство vs. Внешний фокус, дифференциация, соперничество.

Пересечение этих двух осей и образует четыре основные организационные культуры, представленные на рис. 7.34.

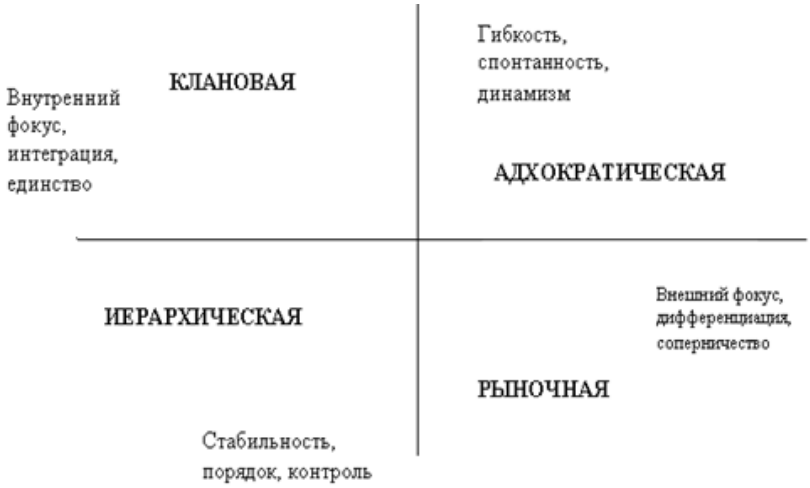


Рисунок 7.34. — Основные типы организационных культур.

В модели «Лидерство» мы можем создать условия для формирования организаций всех четырех типов.

Рыночная (авторитарная) культура задается в модели сильным падением харизматичности от лидера к его последователям. Согласно концепции Вебера, харизматичность лидера в данной культуре максимальна. Лидер обладает максимальной полнотой власти и избегает делегировать ее подчиненным, решая все вопросы сам. В результате подавляющее количество подчиненных оказываются «завязаны» непосредственно на него и лишь несколько из них способны образовывать собственные подчинения. В реальных организациях это зачастую ведет к отсутствию управленческого опыта у подчиненных. При устранении лидера такой организации (например, при его уходе на пенсию) велика вероятность ее распада из-за отсутствия собственных связей между членами организации. Условия формирования авторитарной культуры в модели «Лидерство» представлены на рис. 7.35.

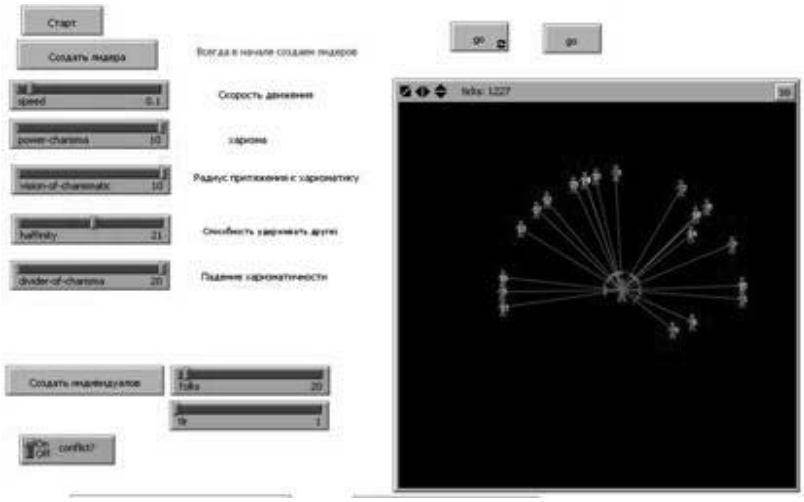


Рисунок 7.35. — Условия формирование рыночной культуры

Клановая культура задается в модели «Лидерство» слабым падением харизматичности при передаче свойств от лидера его последователям. В этой культуре способность членов организации образовывать связи близка этой способности у лидера и каждый из них, включая самых отдаленных от него, могут образовывать связи примерно такой же силы, как и он сам. Некоторые из номинально подчиненных членов могут иметь такое же количество связей, что и сам лидер. Таким образом, лидерство здесь носит скорее символический характер. Организация представляет собой клан – организацию равноправных членов, как правило, существующую в ситуации враждебного окружения и потому постоянно готовую к действию. Устранение лидера в такой организации никогда не ведет к распаду организации. Его место занимает один из ближайших к нему или, что более вероятно, наиболее властных (имеющих большее количество связей и подчинений) членов организации. Условия формирования клановой культуры в модели «Лидерство» представлены на рис. 7.36.

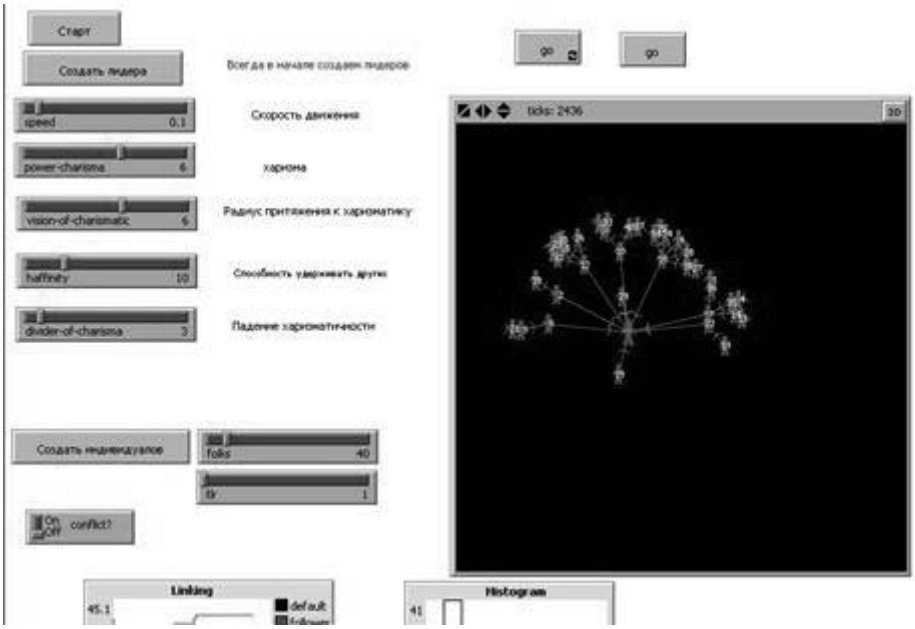


Рисунок 7.36. — Условия формирование клановой культуры

Иерархическая культура задается в модели «Лидерство» сохранением постоянного (хотя и не такого высокого, как в рыночной модели) уровня харизматичности на всех уровнях организации. В такой организации лидер выглядит так же, как и другие члены организации — он обезличен. Номинально он находится в центре организации, фактически есть члены с большим количеством связей, чем у него. Иерархия в отдельных случаях доходит до 15-й степени. Смена лидера в такой модели происходит достаточно безболезненно – ведь важна не его личность, а бесперебойность функционирования системы – то есть чтобы полнота полномочий доходила до самых удаленных членов иерархии. Условия формирования иерархической культуры в модели «Лидерство» (рис. 7.37).

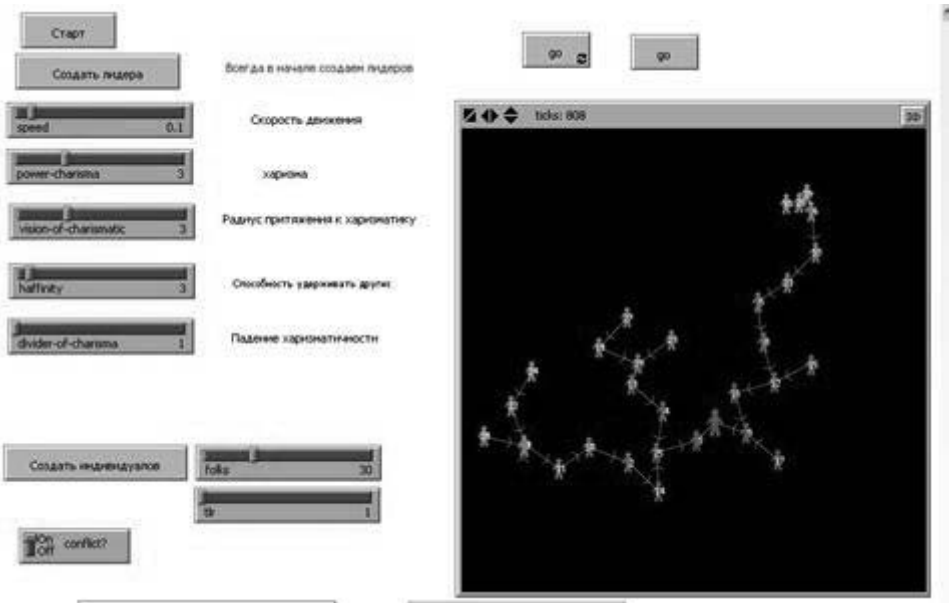


Рисунок 7.37. — Условия формирование иерархической культуры

Адхократическая культура задается в модели «Лидерство» минимальной харизмой лидера в сочетании с его максимальной готовностью делиться властными полномочиями с другими членами команды (дистанция власти = 1). Описание адхократической культуры было сделано американским футурологом Э. Тоффлером спустя полвека после публикации работ Вебера. Он называл адхократию «организацией будущего» и считал, что именно на основе этого типа организационных отношений будет построено будущее общество. В отличие от промышленных гигантов прошлого, адхократические организации невелики, они работают над достижением конкретных, как правило, творческих задач. Они мобильны и могут с легкостью обмениваться между собой своими членами. Уже в наше время мы видим многочисленные организации – стартапы (прежде всего, в IT промышленности), действующие в рамках этой культуры. Условия формирования адхократической культуры в модели «Лидерство» представлены на рис. 7.38.

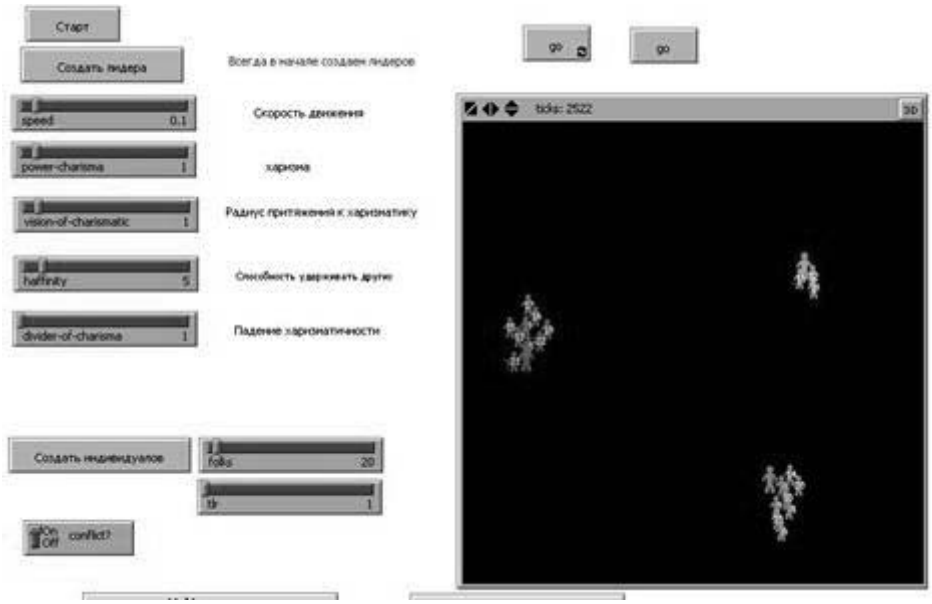


Рисунок 7.38. — Условия формирование адхократической культуры

Столкновение культур. Модель позволяет воссоздавать ситуацию столкновения двух культур и борьбы между ними за «человеческий ресурс» – то есть присоединение максимально большого количества членов. Рассмотрим ситуацию такого столкновения, за которым стоит конфликт организационных принципов, на конкретном историческом примере, известном как «Стрелецкий бунт 1698 года». Произошло оно как раз в то время, когда в России, по инициативе царя Петра, происходил переход от одной организационной модели к другой. Это, в частности, хорошо было видно на примере воинских формирований.

До Петра основу армии составляли стрелецкие полки. Этот вид войска, сформировавшийся еще при Иване Грозном, представлял собой сословную организацию кланового типа. За свою военную службу стрельцы получали значительные пошлинные льготы (у многих из них были свои лавки и промыслы), часто профессия стрельца переходила от отца к сыну. Ко второй половине VII века стрельцы стали представлять собой достаточно грозную и самостоятельную политическую силу, от которой могло зависеть очень многое — вплоть до выбора царя. Петра, стремившегося ввести в стране жесткую вертикаль власти, такая ситуация не устраивала. Он начал создавать регулярную армию, солдаты которой получали жалование, но не пользовались какими-либо сословными преимуществами (иерархическая модель). Естественно, стрельцов, которые усматривали в этом ущемление своих традиционных прав, такая картина не устраивала.

Наивысшей точки конфликт достиг летом 1698 года. Четыре стрелецких полка отказались подчиняться Петру и двинулись из пограничных районов, где они проходили службу, к Москве, для того, чтобы возвести на трон его сводную сестру – Софью. Навстречу им правительством были выдвинуты полки новой регулярной армии под руководством генералиссимуса А. Шеина.

Столкновение произошло у стен Воскресенского монастыря. В нашей модели ситуация противостояния изображена на рис 7.39.

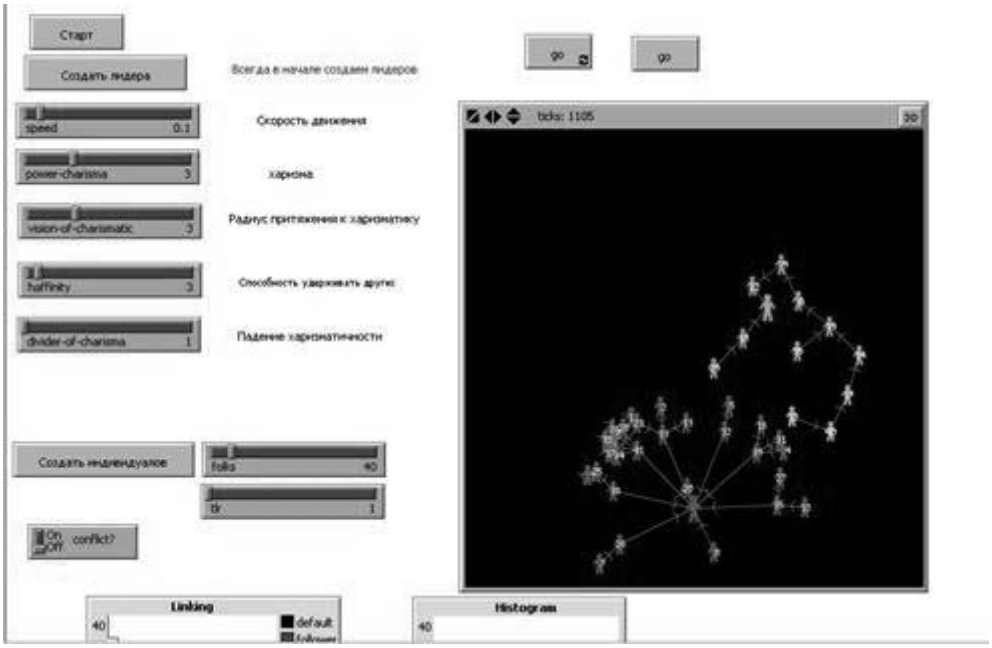


Рисунок 7.39 — Столкновение различных организаций

Как мы видим, стрелецкое войско (красные) более многочисленно. В нем даже выше степень харизматичности (5). Как мы видим, в нем нет высокой иерархии, но зато высока «кучность», характерная для клановой модели. Регулярная армия (зеленые) менее многочисленна, в ней ниже харизма (3), но ее характеризует высокая иерархическая организованность (мы видим связи восьмого уровня, тогда, как в стрелецком войске максимальные связи – третьего уровня).

Для того чтобы развязать себе руки, стрельцы устраняют своих полковников, подозревая их в лояльности правительству (на языке нашей модели стрельцы сделали запрос ask charismatic 0 [ask in-link-neighbors [die]]).

После устранения полковников организационная структура приобрела вид рис. 7.40.

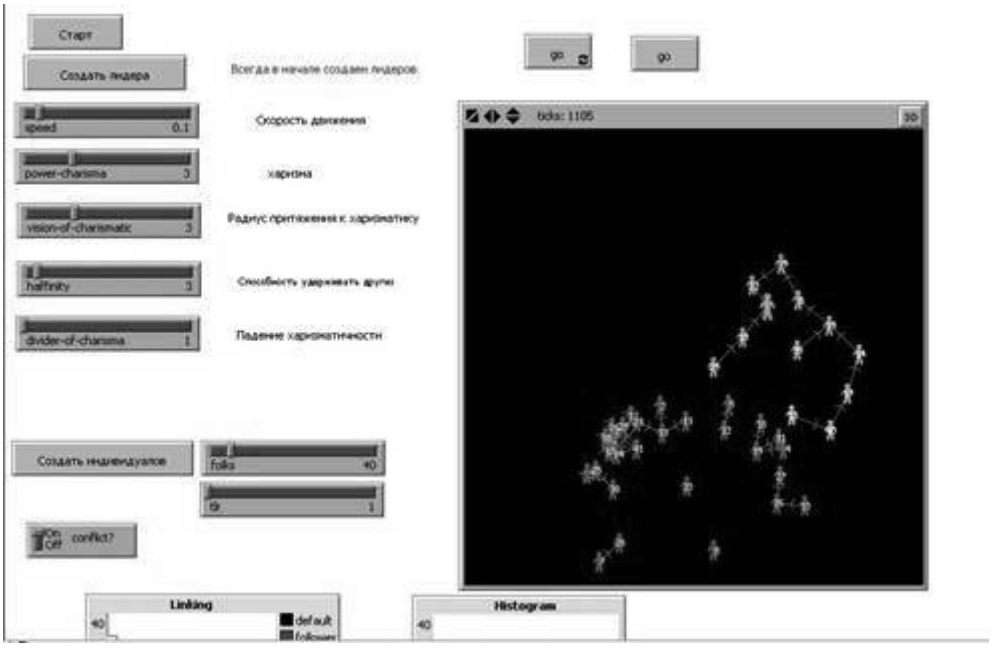


Рисунок 7.40. — Устранение ключевых узлов

Как мы видим, стрелецкое войско остается более многочисленным, но многие связи в нем нарушены. О том, чем это ему грозит, мы узнаем, передвинув рычажок «Конфликт» в состояние «On» (см. рисунок 7.41).

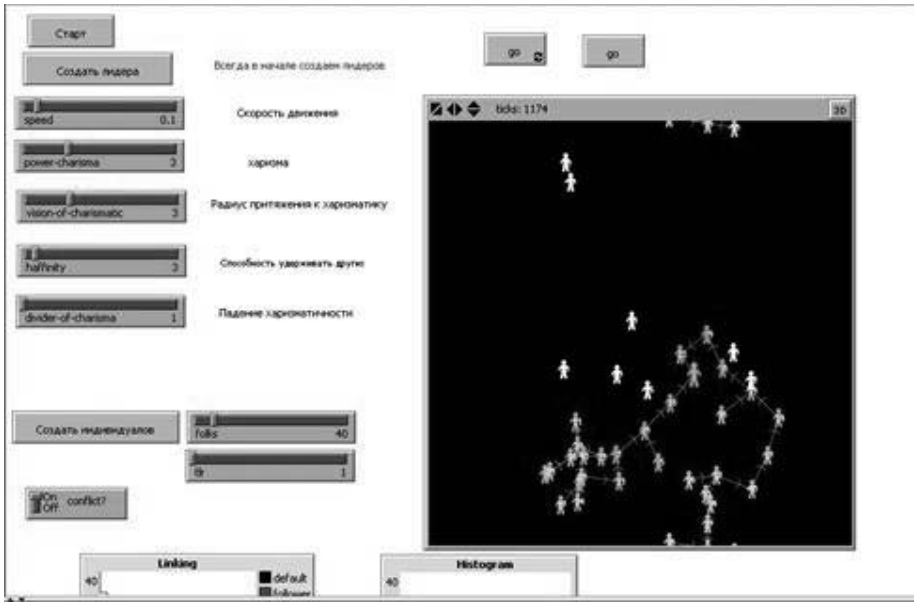


Рисунок 7.41 — Распад группы в связи с потерей руководителя

Когда доходит до дела, оказывается, что стрелецкое войско вчистую проигрывает правительственной армии. Часть стрельцов рассеяна, часть – взята в плен регулярными войсками. Вот как описал эту сцену в своем романе А. Н. Толстой: «... батарея ударила ядрами по обозу, – полетели щепы, забились лошади. Стрельцы отвечали ружейными залпами и бомбами из четырех пушек. В третий раз с холма выстрелили в самую гущу полков. Часть стрельцов кинулась к рогаткам и дефилеям, но там их встретили бутырцы и лефортовцы. Четвертый раз прогрохотали орудия, густым дымом окутался холм. Стрелецкие роты смешались, закрутились, побежали. Бросая знамена, оружие, кафтаны, шапки, драли кто куда. Драгуны, переправившись через речку, поскакали в угон, сгоняя бегущих, как собаки стадо, назад в обоз».

Как мы видим, более многочисленная и даже более харизматичная структура оказывается побеждена структурой с более организованными связями. Фрагмент литературного текста описывающего уничтожение организационной структуры стрелецкого войска внутри модели «Лидерство» представлен следующими командами:

```
ask charismatics with [(my_boss != self) and ((count my-out-links) = 0)] [be_free]
to be_free
ask my-in-links [ask other-end [be_free]] ask my-in-links [die] set breed followers
end
```

Лидеры, потерявшие связь с руководством, распускают своих подчиненных, обрывают свои связи и становятся одиночками.

ВЫВОДЫ

Имитационное моделирование как общий универсальный метод характеризуется следующими достоинствами:

Позволяет решать более сложные задачи;

- Даёт возможность исследовать особенности функционирования реальной системы в разнообразных условиях, включающих критические, аварийные и т.п. (поскольку имитационное моделирование представляет собой машинный аналог (имитацию) сложного процесса, машинный эксперимент с имитационной моделью).
- Существенно сокращает стоимость и продолжительность испытаний по сравнению с натурным экспериментом, с физическим моделированием, то есть экономит ресурсы.
- Позволяет включать ркзультаты натурных испытаний компонентов реальной системы.
- Позволяет достигать лучшие решения за счёт гибкости и легкости варьирования структуры, алгоритмов и параметров.
- Является единственным практически реализуемым методом для исследования сложных систем.

В качестве относительного недостатка имитационного моделирования отметим, что каждое решение носит частный характер, так как оно соответствует фиксированным элементам структуры, алгоритмам, значениям параметров — требуется многократное повторение имитационного эксперимента при вариации исходных данных. Несмотря на принципиальные различия, граница между цифровыми моделями во многом условна, так как все они используют математические модели и вычичлительные процедуры. Другими словами, математические модели представляют одну из важнейших основ имитации.

Имитационное моделирование позволяет решать и такие задачи, как выбор структуры, оценка влияния различных параметров, что составляет основу САПР.

Говоря об имитационном моделировании, необходимо указать на такую важную проблему, как искусственный интеллект. Речь идёт об имитации различных процессов, присущих творческой деятельности человека. Дело не только в том, что элементы искусственного интеллекта необходимо включать в САПР для решения проектно – конструкторских задач. Развитие методов имитации существенно для развития теории и практики искусственного интеллекта как науки, так как основной задачей искусственного интеллекта является задача имитации метапроцедур – процедур универсального творческого характера, когда приходится иметь дело дело со знаниями, а не только с данными. Другими словами без имитационного моделирования трудно познать метапроцедуры при решении интеллектуальных задач.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем назначение стейтчартов?
2. Для чего применяются динамические значения параметров в окне презентации?
3. Как запустить модель на выполнение?
4. В чем назначение слайдеров?
5. Что такое виртуальное время?
6. Как переключиться из режима виртуального времени в реальное?
7. Понятия статистического и имитационного моделирования.
8. Основное достоинство имитационного моделирования.
9. Недостатки имитационного моделирования.
10. Основные процедуры имитационного моделирования

11. Какие классы активных объектов включает проект?
12. Как изменить текущие значения переменных и параметров модели при ее выполнении?
13. Для чего применяются динамические значения параметров в окне презентации?
14 Как переключиться из режима виртуального времени в реальное?
15. Как изменить скорость выполнения модели?
16. В чем смысл эксперимента в программе *AnyLogic*?
17. Какие типы экспериментов поддерживаются программой *AnyLogic*?
18. Понятие «имитационного моделирования»?
19. Общие свойства имитационного моделирования?
20. Как вы понимаете детерминированное моделирование?
21. Какие классы активных объектов включает проект balls?
22. Для чего применяются динамические значения параметров в окне презентации?
23. Как запустить модель на выполнение?
24. В чем назначение слайдеров?
25. Как переключиться из режима виртуального времени в реальное?
26. Понятия статистического и имитационного моделирования.
27. Основное достоинство имитационного моделирования.
28. Недостатки имитационного моделирования.
39. Основные процедуры имитационного моделирования.
30. По какому принципу осуществляется продвижение модельного времени в имитационной модели.
31. Классификация генераторов случайных величин в зависимости от способа их реализации.

ЗАКЛЮЧЕНИЕ

Основным недостатком аналитических моделей является то, что они неизбежно требуют каких-то допущений. Приемлемость этих допущений далеко не всегда может быть оценена без контрольных расчетов. Статистические модели не требуют серьезных допущений и упрощений. В принципе, в статистической модели подходит, что угодно — любые законы распределения, любая сложность системы, множественность ее состояний. Главный же недостаток статистических моделей — их громоздкость и трудоемкость. Огромное число реализации, необходимое для нахождения искомых параметров с приемлемой точностью, требует большого расхода машинного времени. Кроме того, результаты статистического моделирования гораздо труднее осмыслить, чем расчеты по аналитическим моделям, и соответственно труднее оптимизировать решение (его приходится «нащупывать» вслепую). Правильное сочетание аналитических и статистических методов в исследовании операций — дело искусства, чутья и опыта исследователя. Нередко аналитическими методами удается описать какие-то «подсистемы», выделяемые в большой системе, а затем из таких моделей, как из «кирпичиков», строить здание большой, сложной модели.

Весьма сложно четко определить области использования тех или иных методов моделирования. Можно лишь привести основные тенденции, следование которым остается делом вкуса, а также определяется опытом использования уже знакомых инструментов.

Если сказать, что полигональное моделирование (вкупе с сабдивами) больше подходит для персонажного и органического моделирования, это означает: для моделирования антропоморфных форм и различных «членистоногих» персонажей из животного мира для последующей анимации. Масса персонажей может быть сделана при помощи сплайнов, особенно если модели могут быть представлены небольшим количеством поверхностей, например модели пресмыкающихся.

Для моделирования всяческих механизмов и промышленных форм лучше подойдут, наверное, сплайны — так же, как и для дизайнерских работ. Любители моделировать модные автомобили также обязаны вникнуть в сплайновые методы и инструменты. Архитектурное моделирование активно использует оба подхода — в зависимости от конкретной задачи и типа визуализации.

Любую сплайновую модель можно легко сконвертировать в полигональную сетку, поэтому область использования сплайнов существенно расширяется за счет изготовления «болванок» — для дальнейшей полигональной «доводки». Обратное, к сожалению, невозможно, однако любая полигональная модель легко превращается в сабдив, поэтому области использования полигонов и сабдивов часто совпадают.

Еще раз можно отметить, что с появлением эффективных методов работы с subdivision surfaces области использования полигональных инструментов моделирования существенно расширились, и это стимулировало как появление новых инструментов, так и совершенствование уже имеющихся.

Трехмерное моделирование можно довольно условно определить как процесс создания геометрических форм (поверхностей и кривых) для их последующей анимации и визуализации. Можно также условно выделить два крупных направления в трехмерном моделировании: создание реалистичных моделей для компьютерного дизайна и построение моделей для трехмерной анимации. К моделям для дизайна предъявляются, как правило, повышенные требования с точки зрения реалистичной визуализации и точной передачи форм объектов. Сюда же следует отнести и такое направление, как архитектурное моделирование. Можно, однако, с известной долей цинизма сказать, что для дизайна не так важно, как модель устроена или изготовлена, главное — чтобы она хорошо выглядела. Рассмотренные методы можно использовать в обоих направлениях. Все методики имеют свои достоинства и недостатки, и выбор метода будет зависеть от конкретной задачи.

Знание и применение систем компьютерной математики, технического и имитационного моделирования позволяют модельщикам оперативно выбрать систему моделирования, построить адекватные модели, способы их решения, перейти к полномасштабному исследованию реального явления или процесса на модели, оценить решения моделей и представить поведение и закономерности изучаемого явления.

СПИСОК ИСТОЧНИКОВ ИНФОРМАЦИИ

1. Колесов Ю.Б. Объектно-ориентированное моделирование сложных динамических систем / Ю.Б.Колесов. – Петербург : СПб ГПУ, 2009. – 239 с.
2. Маликов Р. Ф. Основы систем компьютерного моделирования : учеб. пособ. для вузов. / Р. Ф. Маликов - М.: Горячая линия-Телеком, 2008. - 280 с.
3. Бордовский Г. А. Физические основы мате матического моделирования./ Г. А. Бордовский, А. С.Кондратьев, А. Д.Чоудори. - М.: Издательский центр «Академия», 2005. - 320 с.
4. Трусова П.В. Введение в математическое моделирование : учеб. пособ. / П.В.Трусова. - М.: Университетская книга, Логос, 2007. - 440 с.
5. Советов Б. Я. Моделирование систем. Практикум: учеб. пособ./ Б. Я. Советов, С. А. Яковлев. - М: Высшая школа, 2005. - 295 с.
6. Советов Б. Я. Моделирование систем. Учебник для вузов / Б. Я. Советов, С. А. Яковлев. - 3-е изд. – М : Высшая школа, 2001. - 328с.
7. Маликов Р. Ф. Основы математического моделирования: учебн. пособ. для вузов. / Р. Ф. Маликов - М.: Горячая линия-Телеком, 2010. - 280 с.
8. http://ru.wikipedia.org/wiki/Maple
9. http://mapleseven.net
10. http://mathhelpplanet.com
11. Поршнев С. В. Компьютерное моделирование физических процессов в пакете MATLAB : учебн. пособ. для вузов. / С. В. Поршнев - М.: Горячая линия-Телеком, 2011. - 736 с.
12. Дьяконов, В.П. MATLAB 6.5 SP1/7 + Simulink 5/6. Основы применения / В.П. Дьяконов – М.: СОЛОН-Пресс, 2005. – 800 с.
13. ДьяконовВ.П. MATLAB 6.5 SP1/7 + Simulink 5/6. Работа с изображениями и видеопотоками / В.П. Дьяконов. – М.: СОЛОН-Пресс, 2005. – 400 с.
14. Getting Started with MATLAB, pdf документ; пер. с англ. Конюшенко В.В. konushenko@afrodita.phys.msu.su
15. http://mathcad.com.ua
16. Кудрик М. Дизайн интерьеров и архитектурное моделирование для всех. / СПб : Питер, – 2008.
17. Уваров А.С. Проектирование и конструирование электронных устройств. - М.: Юнити-Дана, – 2004.
18. Основы компьютерного проектирования и моделирования РЭС. Лабораторный практикум / П.Б.Абрамов, Л. Б. Афанасьевский, А.Н.Горин, А.Г. Фадин; под ред. проф. А.Г.Фадина. – Воронеж : ВИРЭ, 2002. — 268 с.
19. Борисов Ю.П. Математическое моделирование радиотехнических систем и устройств / Ю.П. Борисов, В.В. Цветнов— М.: Радио и связь, 1985. — 176 с.
20. LabVIEW: практикум по основам измерительных технологий. / В.К. Батоврин, А.С.Бессонов, В.В. Мошкин и др. Мн.: Выш. шк., – 2010.
21. Сениченков Ю.Б. Численное моделирование гибридных систем / Ю.Б. Сениченков, 2004.
22. Моделирование систем в программе VisSim: Справочная система / Н.В. Клиначёв , 2003.
23. Алексеева О.В. Автоматизация проектирования радиоэлектронных средств : учеб. пособ. для студентов ВУЗов. / О.В. Алексеева: под ред. О.В. Алексеева.— М.: Высшая школа, 2000. - 479 с.
24. Фрике К. Вводный курс цифровой электроники / К. Фрике — М: Техносфера, 2003. — 432 с.
25. Кулинич Ю.М. Современная силовая электроника : учеб. пособие / Ю. М. Кулинич. – Хабаровск : Изд-во ДВГУПС, 2006. – 95 с.
26. Божко А.В. Компьютерная графика. / Д.М. Жук, В.Б. Маричев. – М.: МГТУ им.Баумана, 2007. - 392 с.
27. Актуальное моделирование, визуализация и анимация. - СПб. : БХВ – Петербург, 2005. – 456с.
28. 3D MAX 6.0 Windows: пер. с англ. – М.: ДМК Пресс, 2004. – 624 с.
29. Анцыпа В.А. Растровые и векторные графические изображения / В.А. Анцыпа . - Информатика и образование, – 2005.- с.56-62
30. Цыпцын Сергей. Понимая VAYA. / С.Цыпцын. – М.: Арт Хаус медиа, 2007. – С.1428.
31. ru/wikipedia.org/wiki/Autodesk_Maya
32. http://autodeskmaya.livejournal.com
33. Быканова А.Ю. Основы SolidWorks. Построение моделей деталей: учеб.-метод. пособие / А.Ю. Быканова, А.В. Старков. – Владивосток: Изд-во ДВГТУ, 2009. – 120 с.
34. Будасов, Б.В. и др. Строительное черчение: учеб. для вузов / Б.В.Будасов, О.В.Георгиевский, В.П. Каминский – 5-е изд., перераб. и доп. – М.: Стройиздат, 2003. – 456 с.
35. https://ru.wikipedia.org/wiki/AutoCAD
36. 3D – технологии построения чертежа. AutoCAD / А.Л.Хейфец, А.Н. Логиновский, И.В.Буторина, Е.П. Дубовикова – СПб.: БХВ – Петербург, 2005. – 256 с.

37. http://www.autodesk.ru — сайт разработчика программ AutoCAD, Mechanical.
38 Книга '3D моделирование в SketchUp 2015. Учебник-справочник, - 2015.
39. Большаков В. Создание трёхмерных моделей и конструкторской документации в системе КОМПАС-3D : практикум. – СПб.: БХВ – Петербург, 2010. – 496 с.
40. http://www.cadacademy.ru — образовательный сайт в области САПР. http://www.solidworks.ru — сайт поддержки пользователей SolidWorks. http://www.ascon.ru — сайт разработчика K0MITAC-3D. http://www.edu.ascon.ru — сайт «КОМПАС в образовании». http://www.eltech.ru/misc/graph/index.html — сайт каталога с примерами решения учебных задач в системе K0MIIAC-3D.
41 http://www.tflex.ru — сайт разработчика T-FLEX. http://www.isi.cad.ru - все о САПР, PLM, ERP.
42. http://www.sapr.ru — веб-сервер журнала «САПР и графика*. http://www.caduser.ru — сайт пользователей продуктов фирмы Autodesk. http://www.dwgseries.com — сайте бесплатными продуктами, предназначенными для работы с файлами форматов DWG и DXF.
43. Adobe Illustrator CS4. Официальный учебный курс. — Эксмо, 2009. — 512 с. — ISBN 978-5-699-36200-4, ISBN 978-0-321-57378-0.
44. AdobePhotoShop CS - Полезные советы от экспертов.
45. http://photoshop.demiart.ru/book/
46. Лоу А.М., Кельтон В.Д. Имитационное моделирование / А.М.Лоу, В.Д. Кельтон– СПб.: Питер, 2004.– 847 с.
47. Карпов Ю.Г. Имитационное моделирование систем / Ю.Г. Карпов, - 2005.
48. Варжапетян А.Г. Имитационное моделирование на GPSS / А.Г. Варжапетян. - Н: учеб. пособ. - СПб.: ГУАП, 2007. - 384 с.
49. Лычкина Н.Н. Имитационное моделирование экономических процессов / Н.Н. Лычкина, 2005.
50. Kelton W.D., Sadowski R.P., Sadowski D.A. Simulation with Arena McGraw-Hill, Boston, 2002.– 547 p.
51. Замятина О.М. Использование Advanced Process Panel и AdvancedTransferPanel в среде Arena 7.0 для моделирования и анализа сложных систем / О.М. Замятина, Н.Г. Саночкина .– Томск: Изд. ТПУ, 2005 (acs.cctpu.edu.ru/books.shtml).
52. Компания Rockwell Automation, описание пакета Arena.
53. Arena Basic Edition User's Guide. Rockwell Software, 2004. – 82 с.
54. Arena User's Guide. Rockwell Software, 2004. – 142 с.
55. Мультиагентное моделирование в среде NetLogo. Учебники для вузов. Специальная литература, - учеб. пособ.,2005. – 170 с.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

АДАШЕВСЬКА Ірина Юріївна

ІНФОРМАЦІЙНІ СИСТЕМИ КОНСТРУЮВАННЯ ТА МОДЕЛЮВАННЯ ОБ'ЄКТІВ

Навчальний посібник
для студентів и аспірантів спеціальності «Інформаційні технології проектування», студентів технічних спеціальностей, в тому числі для іноземних студентів

Російською мовою

Роботу до друку рекомендував М. А. Погрібний

Редактор О. І. Шпільова

План 2016 р., поз.74
Підписано до друку 23.06. 2016. Формат 60x84 1/16. Папір офсетний.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 12,625.
Наклад 100 прим. Зам. № 06/16-25. Ціна договірна.

Видавничий центр НТУ "ХПІ".
Свідоцтво про державну реєстрацію ДК №3657 від 24.12.2009 р.
61002, Харків, вул. Кирпичова, 21.
Друкарня НТУ "ХПІ". 61002, Харків, вул. Кирпичова, 21.
